CDF/DOC/COMP_UPG/PUBLIC/4759

# CDF Run II Data Handling Resource Management and Software Components

**Walsh Brown, Liz Buckley-Geer, Stephan Lammel, Terry Watts, and Eric Wicklund**

**Sep. 30$^{th}$, 1998**
**version 2.6**

This note describes the software components of the CDF data handling system for Run II. The data handling software falls into three categories: software to manage computing resources; software to spool data from tape to disk and vice versa; and software to locate data and dataset information. The CDF data handling strategy is based on the use of batch systems to manage most computing resources. The resources that are expected to be tight in Run II are tape I/O bandwidth, CPU, and data disk space. CPU and tape I/O bandwidth will be managed by independent queues in the batch system. Data disk space will be under physics group control and cache disks will be re-allocated on a longest unused basis. With the use of ROOT's I/O mechanism data can no longer be read directly from tape. Instead files will be staged to disk with the mt_tools package that was already used by CDF in Run I. Data and dataset information is stored in the CDF Data Catalogue and the disk inventory manager. The CDF Data Catalogue is a file based data catalogue with no event level information.

# 1 Introduction

The next collider run of the Tevatron, Run II, is scheduled for spring of 2000. The CDF experiment plans to record about 1 PByte of data during the two year duration. About half of this data is raw data from the detector. The data presents an over 20 fold increase in data volume compared to the previous data recording, Run I. However, the data handling and analysis challenge is even larger: With the upgrade of the detector and trigger system a significantly higher good event purity is expected. With the growth of the collaboration during the last years and the extended physics potential of the detector an increased analysis activity with respect to Run I is anticipated.

Data will be logged to disk at the experiment in a handful of datastreams. Datastreams are a collections of primary datasets to enable easier access and prioritized processing (and re-processing). All primary datasets are defined based on level 3 trigger information. The datastreams will be send over the network to the Feynman Computer Center (FCC) and spooled there to tape. Each datastream will be split into its primary dataset after production. Physics groups are expected to make secondary and tertiary selections based on level 3 trigger and reconstruction information. The goal is to remove as quickly as possible unwanted information/events so that the dataset can be disk resident. We assume that all secondary and tertiary selections will be well organized and datasets inventoried in the CDF Data Catalogue. Our goal is to share datasets as much as possible.

The hardware architecture of CDF Run II for handling data is discussed in [1]. This note discusses resource management and software aspects.

Data handling in Run I was quite successful, although several areas of potential improvements have been identified. The major area mentioned by the physicists is the management of information about the data. A commercial database management system will be used in Run II to store this information. The idea is to have this data catalogue web accessible so users can easily navigate through the available information. A prototype design of the CDF Data Catalogue exists [2].

Tape access will be very different from Run I. The use of less reliable Exabyte 8200 and 8500 technology in Run I caused physicists great pain. For Run II we plan to eliminate any direct tape access by a user but spool any tape-resident data to disk first. This approach is also necessary since ROOT's I/O mechanism requires direct access files. A simple staging system based on the mt_tools package will be used. The mt_tools package was already used successfully in Run I in both online and production. The package has spooled well over 100 TBytes to and from tape and has undergone a major overhaul during the last months.

CPU and tape I/O resources will be allocated to users through a batch system. There will be no global resource/job optimization. Analysis jobs will maintain status and progress information internally. Both CPU and tape I/O resources will be managed independently using the same scheduling policy. Disk space will be managed through quotas and a disk inventory manager and disk I/O bandwidth should be ample. We believe this to be a viable and very flexible approach for the data analysis of the experiment.

A small scale prototype of the data handling software has been put together during the last two months. It uses the Load Sharing Facility (LSF) as the batch system, a prototype version of the disk management software, the overhauled mt_tools package for tape I/O, and simplified version of the CDF Data Catalogue prototype.

# 2 Design Strategies

The large increase in computing needs for CDF Run II, together with changes in the CDF persistency mechanism, advances in computing technology, and user requests make stretching of the Run I data handling approach not an option. Major changes are required that suggest a re-thinking of the overall design. Inside the given constraints our Run II data handling design has led us to a system similar to Run I. This is not surprising as the basic design strategies underlying both systems are the same:

- There is no global resource optimization but analysis jobs are scheduled individually based on available resources.
- There are physics group specific resources with preferred allocation and management delegated to the physics group.
- Status and progress information of an analysis job is maintained inside each job.
- Support the physicist as much as possible in her/his way of thinking and working and do not impose unnatural procedures.

The data handling software has been designed based those strategies and the following goals in mind:

- Provide rapid data access to new and frequently accessed data;
- To allow convenient access to all data in the system. However, access to infrequently accessed data might have large latency;
- To support as many analyses as possible with the available computing resources;
- To manage resources in a dynamic way (where appropriate) so available resources can be reallocated quickly;
- Try to shorten the turn-around cycle during detector commissioning and debugging of the reconstruction programs;

- Allow easy data export/import to/from remote sites;
- Allow the user to take care of most business her/himself and require as little operational support as possible.

# 3 Overview of Software Components

Figure 1 shows a picture of the software components in the data handling system and how they are layered. At the bottom is the operating system or lowest functionality. At the top of the picture is the user and user API. The software components fall into three categories: CDF Data Catalogue, batch system, and I/O associated components. Each component will be discussed in the following sections in more detail. Here we just give a brief description and some status information. A detailed analysis of the manpower requirements for each component is part of the data handling work breakdown structure and in preparation.

- The Database Management System (DBMS) is at the core of the CDF Data Catalogue. A professional (commercial and/or free-ware) DBMS will be used for this component. We consider it existing, tested, and robust.
- Not all database products conform to the X/Open database standard, ODBC. For Oracle and mSQL there exists a free-ware interface package, iODBC, that provides the standard interface. We would like to see the iODBC package supported by Fermilab.
- The CDF Data Catalogue is a definition of various tables, methods, pre-build queries, etc. on top of iODBC (or directly on top of an ODBC compliant database). A first prototype exist but significantly more work is required here.
- We would like to have a web interface to the CDF Data Catalogue so users can navigate conveniently within the available information. Some DBMSes have a web interface but even in that case significant work remains.
- Luminosity information needs to be managed and stored in the CDF Data Catalogue. This will require some extra software.
- Run condition and quality information will be stored in the CDF Data Catalogue. Most likely additional interfaces will be required to manage this information.
- The batch system is a big and important software component. A product has not been selected at this time. We assume that we have to do at the minimum some enhancement to whatever product is selected. LSF is currently being evaluated. We consider about 80% of the component purchasable/existing. The evaluation, enhancement, testing, and proper configuration will require significant manpower.

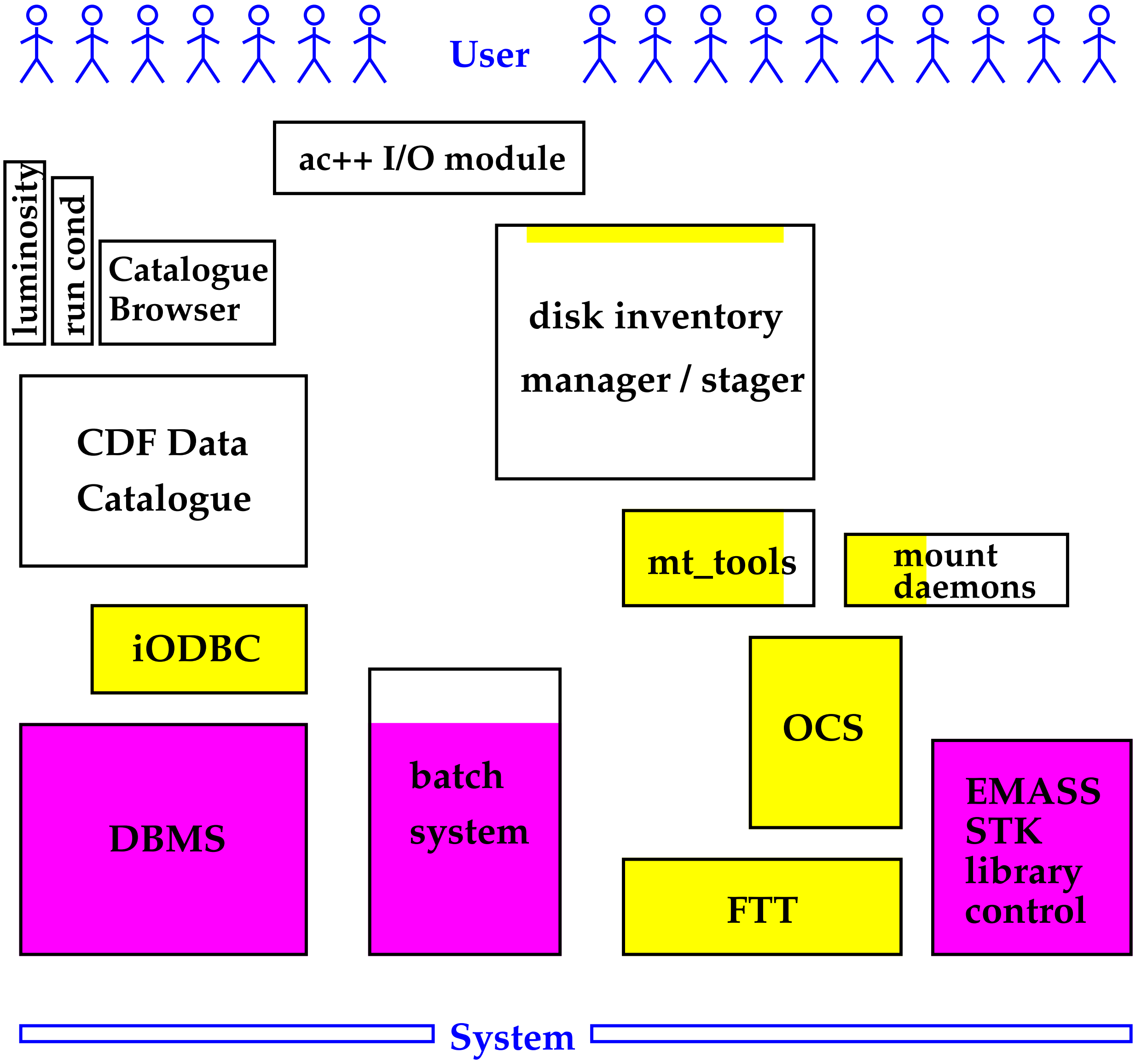


Figure 1: The software components of the CDF Run II data handling system. Magenta (dark grey) indicates existing commercial software and yellow (light grey) existing freeware, Fermilab, or CDF software.

- The FTT package handles all low level tape I/O and device communication. The package exists and is well tested on a large number of platforms and peripherals. The package is maintained by the user application group. The current version is sufficient for our Run II needs. We consider the component existing, tested, and robust.

- The OCS package handles tape drive status tracking, that is allocation, mount status, etc. and tape mount requests. The package exists and meets our requirements. However, the software is hard to install and needs an overhaul to reduce some of its complexity and to streamline the implementation. We would like Fermilab to take on this project. OCS is maintained by the farms and clustered systems group. We consider the component existing, tested, and robust.

- We have library control software for both the STK and EMASS library. The software is provided by the library vendor and updates are available at regular intervals. This is professional software with a high level user API.

- We need to interface the library control software with the OCS package so our mount requests are forwarded to the library controller. There exists an interface for stacker and small scale libraries. The interface to high level STK and EMASS library control software is needed. Including testing, this should be a 1 to 3 months effort for a computing professional. Since we have an interface to small scale libraries we show 1/3 of the component available.

- The mt_tools package is a thin layer on top of FTT and OCS to write, read, label, list, or dump ANSI standard compliant tapes. The package may need some small modification after the tape technology is selected in April of 1999. We show the package about 85% available. It is well tested and robust.

- The disk inventory manager/stager is the largest data handling software component. We are interested in reusing as many components from existing implementation of staging systems. The heterogeneous character of the analysis cluster and our desire to share data disks among the compute nodes requires significant effort on our side. We need at least on full time computing professional working on this component during the next year. The API was thought through in detail during the prototype. The prototype can also serve as a framework for the development of this component, i.e. some of the prototype work will be applicable in the final system (thus the small green bar). We have to consider the software non-existing.

- We need to extend the file input modules of ac++ to handle datasets and filesets. We also need to enhance the output module so it can close files at runsection boundaries. Work on this can start after we settled on the stager design and its API is defined.

# 4 Resource Management

Resource management is the key to successful data handling. In Run I, with inexpensive Exabyte tape technology, I/O resources were plenty. The Run II tape technology will be selected in spring of next year, 1999. The current direction is towards a commodity tape technology and would not impose severe restrictions on the available/affordable tape I/O bandwidth. However, the number of tape drives that can be serviced in the automated tape library depends on the amount of data read (or written) to each tape per tape mount. As long as full tapes are read/written the tape library will not impose restrictions. In the case of filebased tape access/mount the library will limit tape I/O and probably make it the tightest resource.

CPU needs to be managed better than in Run I. On CDF's main UNIX analysis system, cdfsga, the Network Queueing System (NQS) from Sterling Software is used. The scheduling of this batch system was based on a first come first serve basis. Flooding of the system by single users has been the most recognized result. The VMS batch system was enhanced by CDF to handle special resource needs of job. This system too had a rather simple scheduling policy but was able to limit the number of executing jobs per user. We hope to provide sufficient CPU in the central analysis system for data analysis, but Monte Carlo generation and simulation jobs are always hungry for more CPU. We need a convenient way to schedule those jobs to run on simpler (and thus less expensive) computing systems. But even without such jobs the central analysis system will most likely experience congestion at prime times, like at blessings before major physics conferences. Fair CPU scheduling that is transparent to the user can prevent some of the agony during such times. Fair scheduling must take into account the priority of an analysis activity, the amount of CPU already consumed by the user and/or physics group/activity, the number of active analysis jobs in the system for this user/group/activity, and the amount of CPU required to complete the submitted job. The batch system must further be able to handle resources beyond CPU to run, for instance, jobs requiring specific resources at a machine that has those resources available.

Tape I/O and CPU needs are not decoupled. An analysis job requires not only certain mixture of both but requires them at the same time and on the same machine. However, we have decided to manage each of the two resources individually and decouple them as much as possible. This is possible because analysis jobs can no longer read and write tapes directly. This matches our interest to never have an analysis job deal directly with a tape drive. Tape resident data will be spooled to disk by staging jobs and read from disk by the analysis jobs. This approach has several advantages: tape and tape drive failures can be dealt with better and thus make the tape subsystem more reliable; multiple users can access data on the same tape simultaneously; repeated reads of the same tape(s) are satisfied by one tape read (plus multiple disk reads); and it breaks the tight coupling of CPU and tape I/O. The batch system will have two classes of queue: one class that serves CPU needs and a second that handles tape I/O. Each class will have multiple queues, customized to the needs of the experiment. In the

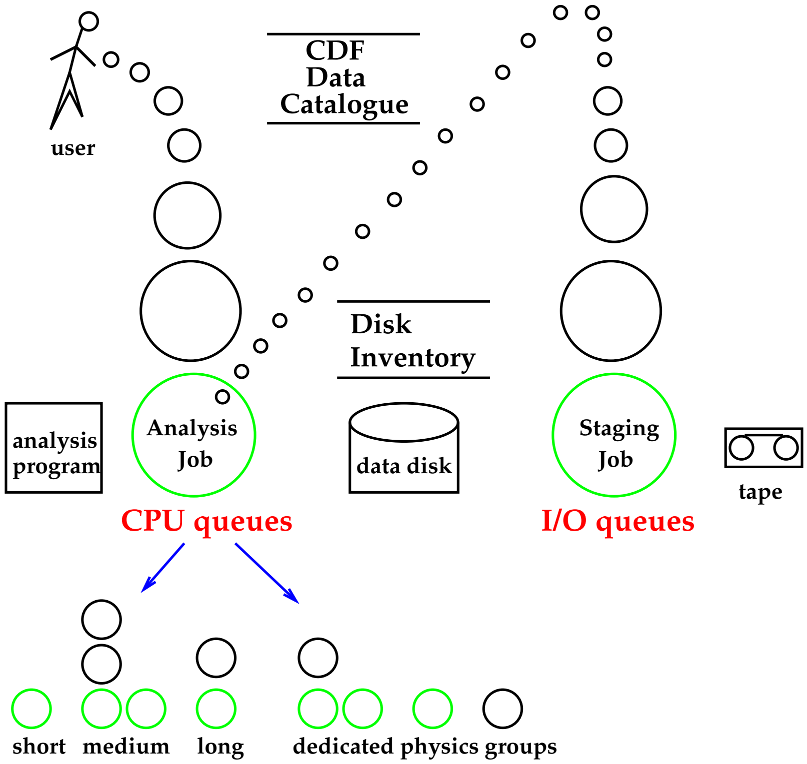

Figure 2: A non-artist's sketch of the CDF Run II batch system.

CPU class there will be queues depending on the amount of CPU required by an analysis job, queues dedicated to physics groups, queues for special analysis activities, etc. The I/O class will have queues for both input and output, for data logging, production input and output, etc. An analysis job may specify the datasets it analyses and automatically launch the necessary staging jobs into the apppropriate I/O queue. The jobs will start execution when the first data of the dataset becomes available on disk. Launching of the staging jobs will be done via a pre-execution step that puts the job into a hold position if no data is disk resident. Upon successful completion the staging job will release the analysis job for execution. The sketch in Figure 2 shows the basic idea of this loosely coupled resource management approach.

Storage space is the third major computing resource that needs management. Our Run I strategy is still the most appropriate approach here. Tape space was managed statically through an administrator (and implemented to a large extent by a web interface). This approach satisfies all our Run II needs and we will continue it. User disk space was assigned a quota and management left to each user her/himself. We plan to take the same approach in Run II with quotas on user login areas in the several hundred MByte range. Scratch space in Run I was managed by quota and a blockdays program that kept the oversubscribed disk from running out of space. With the central analysis system becoming a cluster we will have both local scratch space on each node and spool space that will be cluster wide scratch space. Both spaces will be managed analogous to the Run I scratch space with quota and a daemon preventing the disks to run out of space. Files from executing batch jobs will not be touched. Data disk space in Run I fell into two categories: staging space and physics disk space. For Run II we would like to manage most of the data disk space dynamically via the disk manager/stager system. Physics groups are encouraged to inventory their secondary and tertiary selections in the CDF data catalogue so they are accessible via the disk manager/stager system. Frequently used datasets can be fixed onto disk analogous to most physics disk space usage in Run I (but with the benefit of easier dataset sharing). Staging space will have effectively a management through the I/O queues. Space will be recycled based on total and last access, i.e. longest unused, first out.

# 5 Tape I/O

In Run I CDF had basically three tape I/O approaches/systems. 1) The analysis driver had an interface to read/write tapes directly. The interface was based on the Fermilab packages Operator Communication Software (OCS), Raw Block I/O (rbio), and Fermilab Tape Tools (FTT)[1]. 2) For spooling data from disk to tape and vice versa the mt_tools package was available. The package was used extensively in both online and production, i.e. where staging disks were used. The software package was written between Run Ia and Ib, mainly to address reliability of tape output in production. 3) For data in the Storage Tek (STK) silo the FATMEN/shift software from CERN was used for all I/O.

For Run II we would like to streamline the tape I/O software in CDF. The analysis job driver will no longer access tapes directly but all data will be spooled to disk first. The mt_tools package was written for exactly this task. Run I has proven it to be reliable and up to now well over 100 TBytes of data have been spooled to/from tape by the package. A less reliable script layer included tape mounting and device manipulation commands as well as a primitive scheduling system.

The package has undergone a major overhaul during the summer. A small source of trouble in Run I was traced to the use of the rbio package. This rather heavy package is no longer used. The mt_tools package is now a thin layer on top of the FTT package. The package itself was also extended from pure tape I/O to handle device manipulation, tape mount, and drive allocate through an interface with OCS, thus eliminating the unreliable script layer of Run I. Scheduling of staging jobs in Run II will be done by the batch system,thus making the primitive Run I scripts obsolete.

The design of the Run II data handling system calls for a volume based tape subsystem.

[1] On cdfsga the tape handling software was frozen after Run Ib and FTT based tape handling never installed.

The decision for a volume based approach is based on the the fact that tape volumes are the most natural unit in dealing with tapes. Physics datasets (primary, secondary, and even tertiary) are all much larger than the size of individual tapes. Writing the files of the same dataset to the same physical tape is most natural. If a dataset is read, all files of that dataset are needed and the whole tape is required. Spooling full tapes to disk allows an efficient use of the tape drive resources. The effective I/O bandwith of a tape drive in single file pick or write mode is degraded by as much as a factor of 2 over a volume based read/write even at the 1 GByte file size level [3] (i.e. one requires twice as many tape drives). In order to overcome this problem, requests for files on the same tape would have to be clustered together. The CDF Data Catalogue has to keep track of the file layout on tape so that the staging jobs can be organized appropriately and their execution and completion be managed by the batch system. In most cases where tapes go bad the whole tape becomes unreadable, not individual files. In that case, the volume should be flagged unavailable (while it is being recovered), not individual files. The disk manager has to actively manage the staging disk pool. Over 50,000 files will be in this pool at any given time. With a volume based tape management the disk manager would only need to keep track of the one to two order of magnitude less tapes in its staging pool. Communication between analysis jobs and the disk manager to exchange reservation/release information will be reduced by the same orders of magnitude and are at a comfortable reservation/release per several seconds level.

The OCS package was extended with a small mount daemon that services tape stackers and small libraries. For Run II a similar daemon to forward mount requests to the EMASS and STK library has to be developed. The large libraries come with extensive high level software that should make such a daemon simpler than the one for small stackers where the daemon needs to keep track of the tape inventory itself.

The mt_tools and FTT packages are easy to install also at remote sites. The OCS package, on the other hand, is rather complex and requires some experience. The package has been installed successfully at several remote sites in Run I. However, an overhaul and streamlining of the package for Run II would be most desirable.

# 6 Data Information

The main location of data information is the CDF Data Catalogue. Its design is described in detail in [2]. The CDF Data Catalog will be built on top of a professional database management system (DBMS). The meta data will be organized into a hierarchy of tables containing dataset, fileset (or tape volume/partition), file, and run section information. A fileset is a collection of files located in the same partition on the same tape volume. Tape partitions are a feature found in most modern tape technologies. They allow to partition a physical tape into multiple logical tapes and to add and delete inside each partition, i.e. removing some of the sequential character. Some technologies allow load and unload at begin and end of each partition which is an interesting feature in event-pick mode. Secondary tables will keep processing and filter/selection characteristics that are common among various datasets. The CDF Data Catalogue will also store information on user allocated tapes similar to the CDF tape database of Run I and other information not discussed here. However, it will not store calibration constants, or event level entries.

A second source of data information is the disk inventory/stager. The stager will keep track of all filesets (tape volumes) that have been staged to disk and their location on disk. It will accept reservation requests from users for a particular fileset(s) plus handle the release of reservations. The stager can also provide a list of files belonging to each fileset that is on disk. For each fileset the stager will keep track of the fileset's status: from space being allocated to write a new fileset (tape), to spooling it to tape, staging a fileset to disk, the fileset being used or unused, or the fileset being permanently disk resident. The stager will also provide the interface to the user to launch stage input and output jobs. Those mt_tools based jobs will be submitted into the appropriate batch queues of the system. Filesets can be on tapes requiring different tape drives, for instance, to handle upgrades to a different generation of tape technology or tape drives outside or in a different library. The stager will also be able to handle different staging mechanisms to allow its use at remote sites with, for instance, existing mass storage solutions. To explore and support a heterogeneous cluster the stager system has to know about filesystem formats and what disks are accessible on what compute node inside the cluster. The idea is to keep fileset reservation information locally on each compute node but manage the "mount" count for each disk centrally (i.e. which nodes are currently accessing a particular disk). This will allow us to easily identify orphan reservations and to clean them up.

The disk inventory/stager system will be the front-end to most tape I/O. The FCC based data logger, production, physicists for tertiary/user selections, etc. will all use this interface to spool data to and from disk. It will manage all disk resident data except dedicated physics groups disk space. A prototype of the stager has been put together during the last two months. From the detailed design of the prototype, its implementation, and exercise we have learned quite a bit for its final design/implementation. The stager system was a central piece in our Run II data handling prototype. A separate document [4] describes the data handling prototype and its evaluation in detail.

The disk inventory/stager system will require substantial work during the next six to nine months. While the application programmer interface (API) was worked out rather completely for the prototype, none of the required client/server architecture was developed. We believe this to be the single most manpower intensive task in the data handling project.

# 7 I/O Modules of the Analysis Job

A user submits an analysis job to a CPU queue specifying, for instance, a particular dataset to read. If some parts of the dataset are not yet on disk, those filesets need to be staged onto disk. The analysis job in one of the CPU class queue submits the necessary staging jobs into the appropriate I/O queue. Those jobs are now independent from each other. In the case where no part of the dataset is on disk the analysis job will go into a hold state from where it will be released by the first completing staging job (via a command to the batch system). The analysis job also knows about the launched staging jobs. It will use the information to inform the batch system about how many stage jobs to run in parallel (if resources permit) to keep processing and staging synchronized. Thus, there is a loose coupling between staging and analysis jobs. The use of this loose coupling assumes that we can synchronize the execution of the two jobs within the lifetime of cached data. A quick calculation shows that we should expect disk cache lifetimes on order a day: To replace all 20 TBytes of data disk space in one day we would need over 240 MBytes/sec of tape I/O bandwidth. The planned system will have about 150 MBytes/sec of tape I/O bandwidth.)

The interaction of an analysis job and its related staging job(s) are shown for reading a dataset in Figure 3, and for adding to a new dataset in Figure 4.

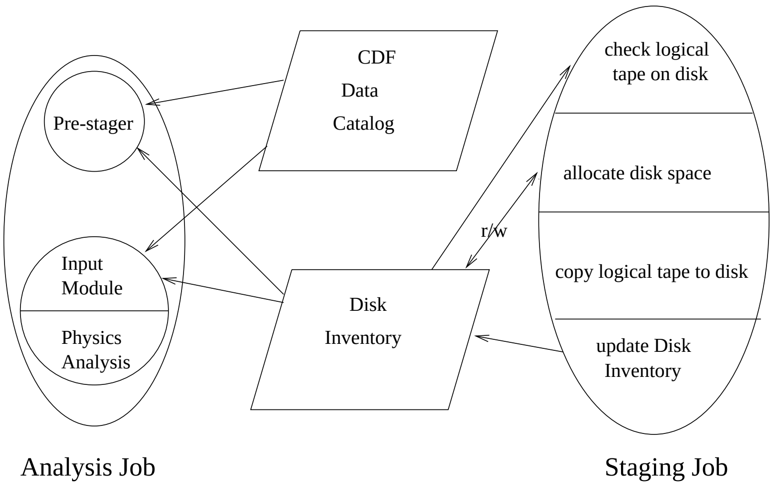


Figure 3: CDF Run II: Job structures when reading a dataset

When reading a dataset, the input module of the analysis job obtains from the CDF Data Catalog a list of filesets. The analysis job checks the list against the disk inventory and launches staging jobs for any filesets that are not disk resident. It then processes the files in each fileset as the fileset becomes available on disk. If no fileset is available after the previous one is analysed the analysis job will merely go into a wait state.

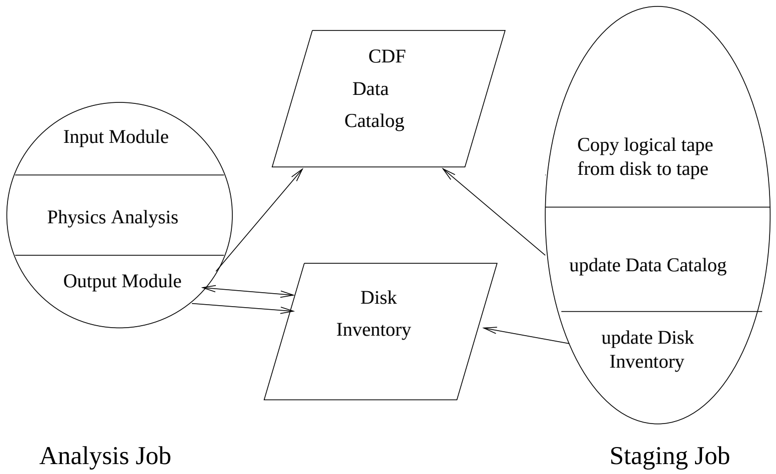


Figure 4: CDF Run II: Job structures when writing a dataset

When writing a new dataset, the output module of the analysis job will allocate space for the new fileset and write the files into the allocated space. Once the fileset is complete it will launch a stage output job to make a tape copy of the fileset. Both the CDF Data Catalog and the disk inventory manager must be updated by both analysis and staging job.

The ac++ input modules will accept filenames as did the analysis_control input module in Run I. A user at a remote site who has no access to the CDF Data Catalogue and no stager software installed can always use this option to read an individual file or list of files. The ac++ input module will now also accept filesets as input. In the case of a fileset or list of filesets, the input module will contact the disk inventory manager/stager to request the filesets being staged to disk and reserve them. Thus, a remote site can have the stager software installed without the CDF Data Catalogue, and this enables processing of data that are larger than the size of the disk in one ac++ job. An additional option of the input module is to specify a dataset. In that case the input module will contact the CDF Data Catalogue to translate the dataset into a list of filesets. The input module will also query file and event numbers to double check files are not truncated or corrupted and the dataset is completely processed.

# 8 What-If and Other Scenarios

In this section we will try to address some of the what–if scenarios put onto the table during the last few month. There must be more such scenarios not considered here. We will try to add them to the list as they are brought to our attention.

## 8.1 Expensive Tape Drives

What if the tape technology selected by the serial media working group whould have high-performance but very expensive ($100k) tape drives? In the case of very few high performance tape drives the direct connection of those drives to the analysis system becomes even more important. IP-network-attaching a 20 MBytes/sec tape drive will either require an expensive, dedicated, high performance network or will degrade the performance so the high performance is only on the price tag. Even with high-performance tape drives we will have at least a few drives for each OS flavour inside the cluster. The primary goal in this scenario is to have disk sharing inside a homogeneous cluster. A stage between the realistic and conservative scenario described in the hardware architecture [1] would do. The stager software keeps already track of reservations and keeping track of mount counts for each disk is already part of the design. We would need to add automatic mount and dismount of filesystems to the stager. With this approach we could share data disks (but with no read access during write) among machines of the same flavour. It is ugly and needs a little investigation, but it should work. The batch system could be easily configured to restrict the staging jobs of an analysis job to the same machine or a machine of the same flavour.

Still, there might be tape drive congestion on one node and an available drive on a machine with different operating system. In this case data have to be send over the network. It is not very desirable but nothing precludes it. The current mt_tools package would have to be expanded with a network I/O module, for instance using the ftp protocol. The extension might be a desirable upgrade in any case so data can be send directly to machines that have no tape drive attached. The manpower for such a network I/O extension is 1 to 3 FTE month.

## 8.2 Enstore

Enstore is being developed for Run II by the High Performance and Parallel Computing department as a replacement for the High Performance Storage System (HPSS) that is installed at Fermilab and used mainly by the KTeV experiment and about five other experiments at any given time. As such Enstore is a network oriented and file based Mass Storage System (MSS). It is not a Hierarchical Storage Management (HSM) System. The system archives files to tape and retrieves them upon request. It keeps track of its file inventory and allows the user convenient access to this inventory as an NFS directory tree via the Perfectly Normal Filesystem (PNFS) developed at DESY. The Enstore system is very different from the tape handling approach described above.

A file based tape subsystem is not desirable for data that have a size much larger than that of a single file (TBytes and hundreds of GByte CDF datasets sizes versus order 1 GByte file sizes) as outlined in sections 4 and 5. We can combine files (for instance through tar or cpio) into large files simulating our filesets and thus reduce Enstore to a volume based system. The chance to hit a 32-Bit (4 GBytes) limit somewhere are high.

Enstore is developed around the idea of network attached tape drives. Reliability, performance, and cost factors make direct attached tape drives desirable as discussed in the previous subsection and Ref. [1]. Sending data via IP over the network is very costly, about 20% of a Pentium-II CPU is required to drive a Fast Ethernet connection at full speed. That would mean one of four CPUs in the analysis system is lost to IP data pumping. Enstore also requires a high performance IP network between all mover and analysis nodes. The developers of Enstore have offered the option of direct attached tape drives but no promise was made to use a direct copy for a local transfer (and not still use IP over the loopback adapter).

Enstore would need to be integrated with the stager which is probably not a big issue. However, if we want to manage computing resources well, we need to know the available resources and must be able to control them. Enstore is designed with its own resource management strategy. Its resource scheduling does not match well our batch based approach and an integration that respects both approaches would have needless complexity. For example, the out-of-order processing of Enstore could make it rather difficult to predict drive availability. As another example, note that volume information is required to organize jobs according to file layout on tape. The volume information of each file needs to be fed back into the CDF Data Catalogue to allow both the analysis job and stager to optimize file request. This duplicate information has to be kept synchronized between the two databases as tape repacking in Enstore can move files between volumes.

Enstore is a large monolithic package. This makes it hard to debug especially since it is written in scripting language. We believe tapes should be written in ANSI standard. Enstore currently encapsulates files into cpio archives and writes those (tape mark separated) to tape. According to the developers it is not intended to be used at remote institutions. Thus, the use of Enstore in CDF would not eliminate the need for an mt_tools based tape I/O system. It would add a second, more complex system (assuming CDF does not want to give up tape analysis at remote institutions). Enstore is still in the development phase. Significant involvement from CDF is needed to push for additional features that would make it a more usable system for us. In addition to this effort, extra manpower is needed to integrate the system with the rest of the Run II data handling.

Maintainance of drive and tape library are always mentioned as Enstore advantage. Fermilab has a maintenance contract with EMASS for both tape drives and library. The customer support group of the Computing Services department logs and dispatches all vendor maintenance engineers. Having an additional hub between system manager (verifying the tape drive problem) and customer support seems like a disadvantage.

## 8.3 Transparent HSM

What about a transparent Hierarchical Storage Manager (HSM)? We could use such a software package very well to assign a kind of user data space. For private user data, N-tuple, logfiles, etc. a transparent HSM would be ideal. Data is no longer of the size of a tape but at file size. A transparent HSM could stretch the user space beyond the size of available disk space by automatically migrating files to tape but keeping the directory entry, i.e. the file visible to the user. The HSM software will intercept any read request to a migrated file and restore it without user intervention.

Transparent HSMs have not been too successful in the past. The VMS Hierarchy package on FNALD was a transparent HSM. Since then next generations of such software packages have become available. Consequently, while the stager with user allocated tapes satisfies our Run II needs, we should keep our eyes and mind open to new attractive HSM solutions.

## 8.4 Disk Failures

What happens if a disk crashes, how are the stage inventory and CDF Data Catalogue updated? The CDF Data Catalogue keeps no information on disk residency and location of a dataset, fileset, or file. The stager system has a list of available disks in its configuration file. The bad disk has to be disabled in the configuration file and any daemons refreshed. In the case of a daemon providing mount service the daemon itself could recognize a disk failure and trigger the reconfiguration. How automated and smooth disk failure are depend on the effort CDF puts into the development of the stager software. The reader should keep in mind that a large fraction of the data disk space (or even all) might be RAID protected in which case a single disk failure would not result in a data loss.

## 8.5 Overlapping Staging Requests

What about multiple users submitting staging requests for the same fileset or dataset? We hope this will occur frequently as that means datasets are being shared. The staging job allocated disk space before it starts the actual spooling to disk. The disk space is allocated under the fileset name and in the cases when the fileset is already disk resident, or in the process of being staged to disk, etc. the stager will not allow a duplicate allocation. The second stage input job simply terminates and both users will wait for the first staging job to complete. Since the analysis jobs check only with the disk inventory manager they are not concerned which stage input does the actual spooling. Since the two users now share the I/O activity, each user has now effectively double I/O resources available (assuming the analysis job can process the data that fast, otherwise the jobs will throttle back the I/O bandwidth).

## 8.6 Files in Specific Order

How do I process the files of a dataset in a specific order? If a dataset is specified as input the filesets belonging to the dataset will not necessarily be processed in the same order in which the CDF Data Catalogue lists them. Any filesets of that dataset that are already on disk will be proccessed first. The files inside each fileset, however, will always be processed in the same order. To process all files of a dataset (or data-subset) in a user defined order, the user would use multiple input commands inside ac++, one for each fileset (analogous to the current analysis_control).

## 8.7 Remote Data Access and Analysis

How do I analyse data at a home institution? There are a large variety of options here. Let's discuss the two extreme cases. In the thin client approach, a user at a remote site has an X-terminal and connects to a compute node in the central analysis cluster. She/he does all her/his work on the system located at Fermilab. She/he never exports/imports data, installs CDF specific software, etc. The approach requires a low latency network connection to Fermilab from the remote institution.

The second extreme case is where all CDF software is installed at the remote site. In this case datasets (or individual filesets) can be imported at the remote site either via the network (small amount of data) or tape copy at Fermilab and shipment. Data import via the network would be a simple ftp from one of the compute nodes in the central analysis cluster. For data import via tape, image copies should be made so the volume identifier matches the original tapes. In this case the entries in the CDF Data Catalogue are valid and can be used directly. The imported data can be used by specifying filenames, filesets, or datasets. (Please see the more detailed discussion under *I/O Modules of the Analysis Job*.

## 8.8 Datalogger and Production

How will the datalogger and input/output from production work? The FCC datalogger is a simple stage output system. A small batch system that checks the amount of data available for each stream in the B0 disk buffer and launches stage output jobs if appropriate. The B0 datalogger writes to disk. It will use a special output mode to write files close to the preferred filesize (about 1 GByte) as possible. Files will be closed only at runsection boundaries which are defined by the luminosity stream. Events need to be sorted at runsection boundary so that all events belonging to one runsection and datastream are in one file. The handshake with the luminosity stream is not yet defined. For the event sorting at runsection boundary a simple but efficient algorithm exists. The same sorting-at-runsection-boundary-mechanism will be used in production output when datastreams are split into the primary datasets.

The input and output of production depends on the hardware architecture. If we assume that the reconstruction farm has a dedicated I/O server, then it would stage the raw data filesets to disk, copy the files to the processing nodes, and launch the reconstruction jobs there. Upon completion, the reconstruction jobs would return their output to the I/O server where files are either concatenated or staged directly to tape. The output module in each reconstruction job (or the output module of the concatenation job) would update the file information in the CDF Data Catalogue and the stage output job would update the fileset information.

# References


[1] W. Brown *et al.*, CDF Run II Data Handling Central Analysis System Hardware Architecture, CDF note 4707 (1998).

[2] E. Wicklund, CDF Data Catalogue for Data Analysis in CDF Run II, CDF note 4761 (1998).

[3] J. Bakken *et al.*, Serial Media Working Group First Run II Report, CDF note 4560 (1998).

[4] W. Brown *et al.*, CDF Run II Data Handling Prototype Plans, Evaluation, and results, CDF note xxxx (1998), in preparation.