DØ Note 3464 **CD TN87**

System Software and Hardware Architectures for the Sequential Access Model

Mike Diesburg, Dorota Genser, Krzysztof Genser, Lee Lueking,
Frank Nagy, Don Petravick, Ruth Pordes, Kurt Ruthmansdorfer,
Heidi Schellman, Marilyn Schweitzer,
Igor Terekhov, Matt Vranicar, Vicky White

June 12, 1998

Abstract

The Sequential Access Model is a software framework being developed to facilitate data management and access for Run II. The software and hardware architectures for this system are discussed and the functional elements envisioned for the system are described. Examples are provided illustrating how the system might work for various modes of data access which are anticipated.

Contents

1 Introduction **3**

2 System Software Architecture **3**

2.1 Objectives . . . 3
2.2 System Overview . . . 5
2.3 The Components of SAM . . . 8
2.4 File and Event Database . . . 10
2.4.1 Description . . . 10
2.4.2 Data Loss Handling . . . 14
2.4.3 Trigger and Stream Definitions . . . 14
2.4.4 Luminosity Data . . . 14
2.4.5 File Information . . . 14
2.4.6 Derived Dataset Records . . . 15
2.4.7 Summary of information needed for normalization . . . 15
2.5 The Warehouse . . . 15
2.5.1 Storage Management System . . . 15
2.5.2 Managed Disk Cache . . . 15
2.6 Various Modes of Access . . . 16
2.6.1 File Access . . . 16
2.6.2 File Import . . . 17
2.6.3 Event Picking . . . 17
2.7 How The System Might Function . . . 19

3 System Hardware Architecture **21**

3.1 Objectives . . . 21
3.2 Database Server System . . . 21
3.3 The Data Warehouse Components . . . 21
3.4 Example of Various Hardware Configurations . . . 22
3.5 SMP/Enterprise Server . . . 23
3.6 Distributed Network CPU . . . 23
3.7 Off-site Data Delivery . . . 27

4 Summary **27**

A SAM Control and Data Flow **28**

1 Introduction

The Sequential Access Method (SAM) system is intended to serve very large, of order 1 PB, data samples to several hundred simultaneous users. Because shared hardware resources such as robotic arms will be subject to contention, the SAM system cannot merely pipe data to the end user but must be able to set priorities and allocate resources. In the Run II data management scheme, The SAM is both an umbrella and a component piece. From the users' point of view, SAM is the connection between what he wants to do and the data he needs to get it done. SAM is the primary, if not the only, control and navigation tool which will be provided to users with which to access data, and therefore will be responsible for all data deliveries and blamed for all failures. However, from a design point of view, SAM is a small component which brings together the functions of the metadata, Mass Storage, processing farms and analysis systems.

The system software and hardware architectures proposed to meet the requirements as specified elsewhere [1],[2],[3], and [4] is presented. The system has been approached from each a user's, administrator's and maintainer's point of view. It is desired to provide a minimal system which provides the reliability and functionality for the many types of data access anticipated throughout the life of the experiment. The software and possible hardware configurations are being considered together to develop a more comprehensive picture of the problems and the SAM architecture being developed to meet them.

2 System Software Architecture

2.1 Objectives

Conceiving a single system architecture which will meet all of the requirements is one of the major goals of the sequential access method. This will assure that the system is flexible enough to provide all of the access modes, easy to use and readily maintainable. Such a system also provides features, such as a common point request interface which enables central screening and auditing of requests for data, while not interfering with actual data delivery. In addition, the system provides a way for all requests to be pooled together and sorted by tailored algorithms to make maximal use of available hardware resources. With an integrated architecture it is easy to monitor the uses and flows of the system and tune them to meet demands. An integrated approach to organizing information about the system will promote a uniform structure and a single set of common tools to provide bookkeeping for many diverse types of data import and delivery activities.

To bring together all of the ideas needed in such a system we first considered the diversity of requirements. Individual systems were discussed which would satisfy the control and data flow needs for each of the major SAM access patterns: Production Farm, Freight train production processing, On-demand file delivery and Event picking. In the process of evaluating each of these in detail, the control and data flows were identified and documented and some of these are included in Appendix A. Several features were identified which are needed by all systems, some of thes are restating requirements specified in much more detail in the requirements document.

- Submission of "requests" for data by either filenames or data-related (run, trigger, etc.) criteria.
- The concept of a "warehouse" which contains all long-term data storage.
- The concept of a "delivery station" to which data is delivered from the warehouse.

- The concept of a "consumer" who processes the data.
- The concept of a "project" which is a group of needed files or events which can be shared by many consumers.
- Processing multiple files simultaneously by consumers.
- Optimizing tape access in the warehouse.
- Data caching for files to be reused without accessing tape again.
- Efficient access to multiple storage sources in the warehouse.
- Straightforward Graphical User Interface for *administrators* and *users* to interact with the system.
- User Application Program Interface for administrators and users to interact with the system..
- Multi-stage processing tasks which may include merging of files.
- System of bookkeeping to assure completion of projects.
- The ability to provide intermediate status information for any project.
- The ability to provide detailed completion summaries for a project to each consumer.
- Initial estimator for project duration.
- Incremental update estimations for long term projects.
- The ability for a user to register to an existing project.
- Support for many diverse hardware configurations with varied processing objectives.
- A database system which will provide information about what data is in which file, and keep track of data processing steps.
- Different, and levels of authorization for requests for users, groups of users, and administrators.
- Network distributed set of intercommunicating processes.
- Scalable and adaptable to many hardware configurations.

In the sections below, specific examples are given which illustrate the content and behavior of various parts of the system. These details provide a concrete picture of some of the ideas being discussed regarding how the system is envisioned to work, although a detailed design is not completed nor implied.

2.2 System Overview

The architecture which has been developed includes the common elements, and attempts to accommodate the unshared elements, of the identified access modes. The principal functional components are the

1. Request broker,
2. Event and file database,
3. Warehouse,
4. Station(s), and
5. Consumer(s).

All requests submitted to the system go through a *request broker*, which can review the magnitude of resources needed to fulfill the request, and the authorization level of the submiter. Information pertaining to physics attributes, file content, processing details and other parameters is available from the *event and file database* allowing users and the system to interpret requests and select data. The *warehouse* includes all of the elements of the system which enable the physical access to the data including optimization, inventory for each storage device, and the interfaces to the physical storage devices. The station is the destination/source for all data requested-from/sent-to the warehouse, and includes the bookkeeping elements needed to account for the transactions as well as facilities to manage local buffer storage. *Consumers* are the processes which use the data delivered to a station as input and produce processed data as output. The system consists of only one instance of each the *event and file database* and *warehouse*. These common elements are shared by many *stations*. For each station, there are potentially many consumers. These basic components of the system are illustrated in Figure 1.

All requests for data are entered into the system through *request brokers* which consult a central shared database for authorizations and resource allocations. This enables the system to have central control over all submitted requests and direct them to an appropriate processing resources. There are four basic functions of this element of the system

1. receiving requests from users or administrators,
2. estimating the magnitude of the request,
3. checking if the requester has the authorization level needed to continue the request and
4. either sending authorized requests on to an appropriate *station* as a "*project*" or informing the user his request is not authorized.

To estimate the size of a given request, the broker must consult with the *event and file database* through a *translator* to have a general request translated into a list of needed files. It must also get an estimate from the warehouse as to how long delivering these files might take, given factors such as the number of tape mounts and the current load on the warehouse delivery resources. This strategy allows controlling the system, at the request level, and avoiding unauthorized users making unrealistic demands on the system. The actual control parameters in this part of the system are likely to vary significantly throughout the course of the run and might be controlled dynamically to balance the overall system.

The *file and event database* provides the system access to event and file metadata. This database contains trigger and selection information for each event as well as run conditions and

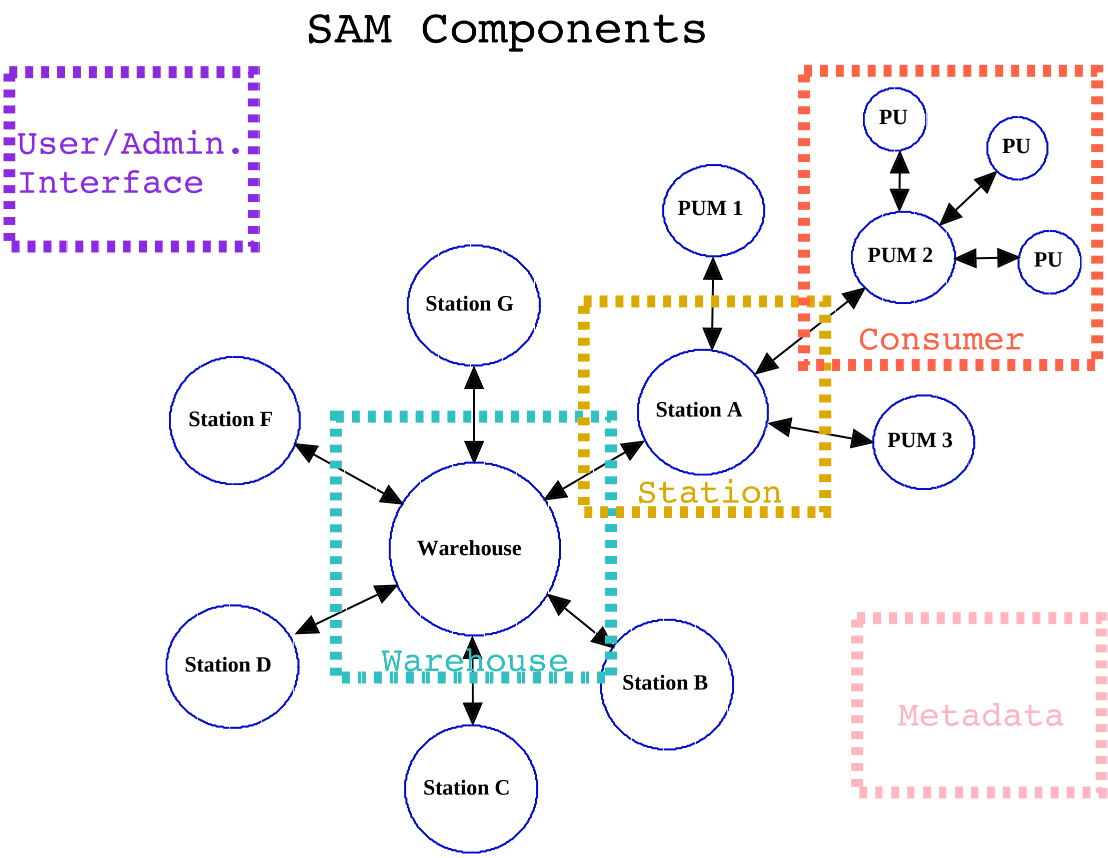


Figure 1: Components of the Sequential access Method.

processing information for each data file. There are two active components which get data to and from the data base: (1)*accountant*, and (2) *translator*. The accountant is an interface to the database which is employed to store information about the data at the file level, record significant processing steps and provide access to all ancillary information. The translator converts requests into files from whatever abstract form in which they were submitted.

At the center of the system is the data *warehouse*. This part of the system includes all functional elements which allow the system to deliver data and accept import data in an efficient manner to and from specified destinations. Included in the warehouse are the interfaces to all long-term data storage repositories such as (1) file disk cache, (2) pick events disk cache, (3) pick events tape cache, and (4) storage manager for automated tape libraries and shelf tape storage. File requests coming from the many delivery stations are merged and sorted by an optimizer which uses rules designed to provide high priority jobs quick data access and minimize the number of tape mounts. As in the *request broker*, these rules may be dynamically varied to meet changing demands on the system. An overall storage manager is included which functions to check the inventory for each physical storage location and sending requests for each file to the proper storage device.

The *station* and *consumers* are closely interrelated. A station has the capability to both accept data from and send data to the warehouse. The roles of the station are to (1) record a status list of files to be delivered/sent, (2) send a list of files to the warehouse for action, (3) monitor which files are currently on the station's buffers, (4) keep track of which consumers have processed each file in the input buffer, and (5) purge the input buffer of files which have been processed by all consumers. This part of the station manager is a type of queue manager, and manages a list of files called a *project*. Multiple *projects* are allowed to accumulate in a station. In order for this to work properly for many consumers interested in the same *project*, a method of registering each consumer with a particular *project* is required and each consumer must inform the station manager when he starts and finishes processing a file. In many cases, a consumer will require multiple parallel processes to consume data at the desired rate and the additional function of managing these must be introduced. Each consumer's data processors are called "Processing Units" (PU) and their coordinating processes referred to as "Processing Unit Managers" (PUM).

An additional pick event function is provided to the station. In this mode the unit of transaction is the event, as opposed to the other modes where it is the file. This changes, slightly, the way the system behaves as projects are now collections of events with their associated file.

The interface to the system is provided through GUI and API. For some activities, a graphical interface will provide an effective way to create and manage projects. This is especially true when the project contains many hundreds or thousands of files. Major processing tasks, such as farm reconstruction and production jobs will employ such an interface to great advantage. A GUI will also enable new users to learn the system quickly and such an interface reduces the number of mistakes a requester might make since it guides him/her down a simplified path. On the other hand, for the user who has a simple request of one or a few files, and for the sophisticated user, a GUI will be quite cumbersome. For these users, API's will be provided which will make the system appear transparent to the user, i.e., when he asks for a file or list of files in his/her application, those files will show up and his application will process them. Features and options will be provided to enable maximum effectiveness of the hardware resources and to avoid many concurrent users grid-locking the system. Whether a task is submitted via GUI or API, any user or administrator will be able to easily monitor the status of any job.

As outlined above, the system is comprised of a few, well defined, modular elements. This allows the system to be distributed among many intercommunicating processes operating in a network-distributed multi-platform hardware environment. This allows the system to be easily scalable and adapt to the many processing environments which we anticipate. In this architecture, new elements can be added at any level without changing the way in which the system operates. For example, a new robotic tape library can be added to the warehouse without changing the way the warehouse and the outlying delivery stations function. Delivery stations are independent of each other and new ones can be added to provide additional processing of any type. Additional processing capability can be added to a consumer, and the interaction with the delivery station will remain the same.

2.3 The Components of SAM

The major components of the system are represented in the data flow diagram of Figure 2. Descriptions for each component are as follows:

1. **Request Broker:** All requests from all users and administrators must go through request brokers in order to allow the system to manage resources on a global scale, and screen for unreasonable requests. This part of the system books requests for projects. It also registers users to projects on particular stations, and gets tokens for consumers to join a project.

2. **Event and File Database:** A common database to the system provides all run conditions and processing information for each file. It also stores trigger and other information for each event, which allows making queries to the database at the event level.

   (a) **Accountant:** Provides utilities to add and retrieve file and event information to and from the database.

   (b) **Translator:** Translates a set of constraints, such as run range, trigger types, etc. into a list of data files.

3. **Data Warehouse:** The *warehouse* deals exclusively in tapes, files and events and knows nothing about the world outside itself. It includes the *optimizer*, a *storage manager* and methods to request data as files or events from any long term storage device.

   (a) **Store Manager:** All long-term storage repositories are contacted through the *storage manager*. The long-term storage areas include PE Hot, PE Warm, Thumbnail, Cache Disk (a.k.a. Slush Disk), Automated Tape Library, Shelf Tapes. The Storage Manager sends requests to the manager for each device to start sending data to the appropriate buffer managers. The storage device establishes communication with the appropriate *buffer manager* and begins sending data.

   (b) **Optimizer:**The optimizer works on the merged requested file list provided by the various *stations* . It uses inventories for each storage device to determine where the fastest place to get any given data is. It then creates a list of the files which are to be provided by each storage device.

4. **Station:**A processing unit consisting of hardware resources and software control over those resources. The hardware includes configuration of network, buffer disk and processing CPU. The software control manages the resources in such a way as to assure maximum data throughput through the system and error recovery.

   (a) **Station Manager:** This part of the station manages resources for each station configuration. The station manager maintains a table in which he keeps track of the files/events required for each project and the status of each consumers file/event usage. As each file/event is delivered to the station, it is marked as “in”, and available for consumers to process it.

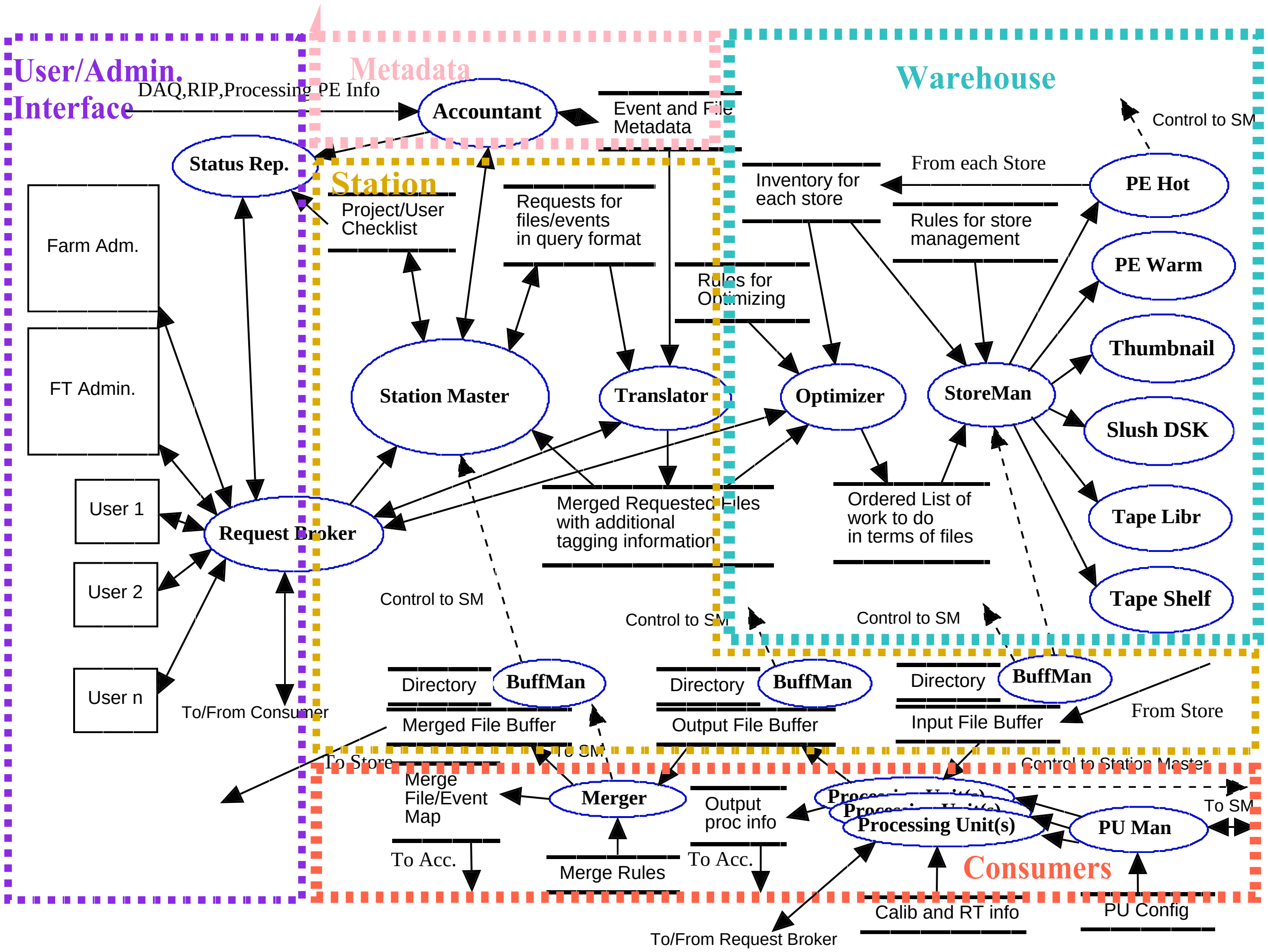


Figure 2: Data Flow diagram for The Sequential access Method.

i. **Delivery Features:** The "start" time is recorded when each consumer begins processing, and the "end" time or "error" condition is recorded when a consumer finishes or fails processing a file.

ii. **Import Features:** Data produced by consumers is sent to the warehouse for import. The station manager assures that the data is managed correctly and that the proper information is sent to the file and event database.

iii. **Pick Features:**When a station is used for event picking, entries are tracked at the event level. The file and tape information for each event is included with the event so that requests from different users can be sorted and grouped together by tape and file.

(b) **Buffer Manager:**

i. **Input:** Data is delivered to the station's input buffer from the warehouse.

ii. **Output:** Data from consumer processing is sent to the output buffer.

iii. **Merge:** In many cases, consumer processing will result in files which are much smaller than the input data files. It is advantageous to provide a mechanism to merge groups of these smaller files into larger files. The merge buffer is used for this processing.

5. **Consumer:** A processing task which consumes data from the *input buffer* in a *station*. Each consumer must be *registered* to a *project* and may have one or more *processing units* acting on the project simultaneously.

(a) **Processing Unit (PU):** The process which runs the executable to process the project data for a consumer. Each processing unit is assigned a separate file by the station master.

(b) **Processing Unit Manager (PUM):** A single process which manages the processing units for each consumer. The action of each processing unit manager is defined by a script provided station master as part of the user registration process. It is the station manager's responsibility to see that this script is running.

(c) **Merging Unit:** A part of the system which concatenates many small files together to make fewer large files, according to some specified rules.

2.4 File and Event Database

2.4.1 Description

The first attempt at a model for the file and event database is given in the Entity-Relationship (ER) diagrams shown in Figures 3 and 4. Descriptions of the various elements follow.

- A Run identifies a set of data acquired sequentially with a single set of calibration and detector constants. A Run is identified by an increasing run number and described by details such as its begin/end time, description, and run conditions.

- Each Run produces many physics Events, which are described by an unique event number and their defining trigger bit settings and event filters. The raw data for each event is stored in the raw data tier with one file containing multiple events. Subsequent processing on the events results in the creation of additional files for the other data tiers: fully reconstructed (EDU250), summary reconstructed (EDU50/EDU150), highly compact summary physics data (EDU5/Thumbnail).

- The numerous Files created for a run each contain information regarding many events. Files are identified by their filename and described by attributes such as an event range, trigger information, sequence in the run, create time, data tier (edu type), format information, file status.
- The Event to File Map provides a means to map related events and files. To reduce the database storage required, alternatives to storing simple minded key entries for each event to file map will be considered. For example, the start and end event ids for a file might define the map. The exact implementation of this mapping technique has yet to be determineed.
- Each file will exist on one or more volumes. A file may reside on a tape volume, and it may also be buffered on disk. While the tape volume may be considered the "source" of the file, during it's lifetime in SAM, a file may exist in multiple locations.
- Each file is classified by a Data Tier, which defines the type of data content in the file. The Data Tier describes the level of consolidation that the file represents: raw data, fully reconstructed (EDU250), summary reconstructed (EDU50/EDU150), highly compact summary physics data (EDU5/Thumbnail), or unknown/other.
- Physics analysis is performed over Streams of events. These Data Streams are distinguished by a particular set of trigger and filter bit setting combinations which are depicted in the Trigger Stream Map. It is expected that there will be about 32 data streams of interest.
- Each Physical Data Stream identifies the files that comprise the stream, whereas a Logical Data Stream provides a means of grouping Physical Data Streams.
- A Dataset consists of a set of data useful from a physics perspective. It may be a set of data constructed by an analysis process, or it may be a set of data used by an analysis process. Physical Datasets identify a particular set of files of interest. Logical Datasets are identified by a description or set of criteria that form the dataset.
- Physics analysis is performed by Analysis Projects. For each Project, the initiating Person, the Project Inputs and the requested list of Project Files is recorded.
- As requested files are processed for a project, the User who is accessing each file will be tracked via a Project File User mapping. As a part of this mapping, the status of that User's file access is recorded, to indicate if they are reading the file, done with the file, etc.
- Processes which either create new or use existing files are run as either User processes or Farm processes. These processes are run on a particular Station.
- For each Process, its Process Family definition will allow for the tracking of which versions of the various different types of online software, reconstruction version, Monte Carlo version, etc. were used.
- User or Farm Processes will either use existing data files or produce additional data files. The consumption or production of files will be recorded in the appropriate File Production Map or File Consumption Map, which allows for process level tracking of file usages.

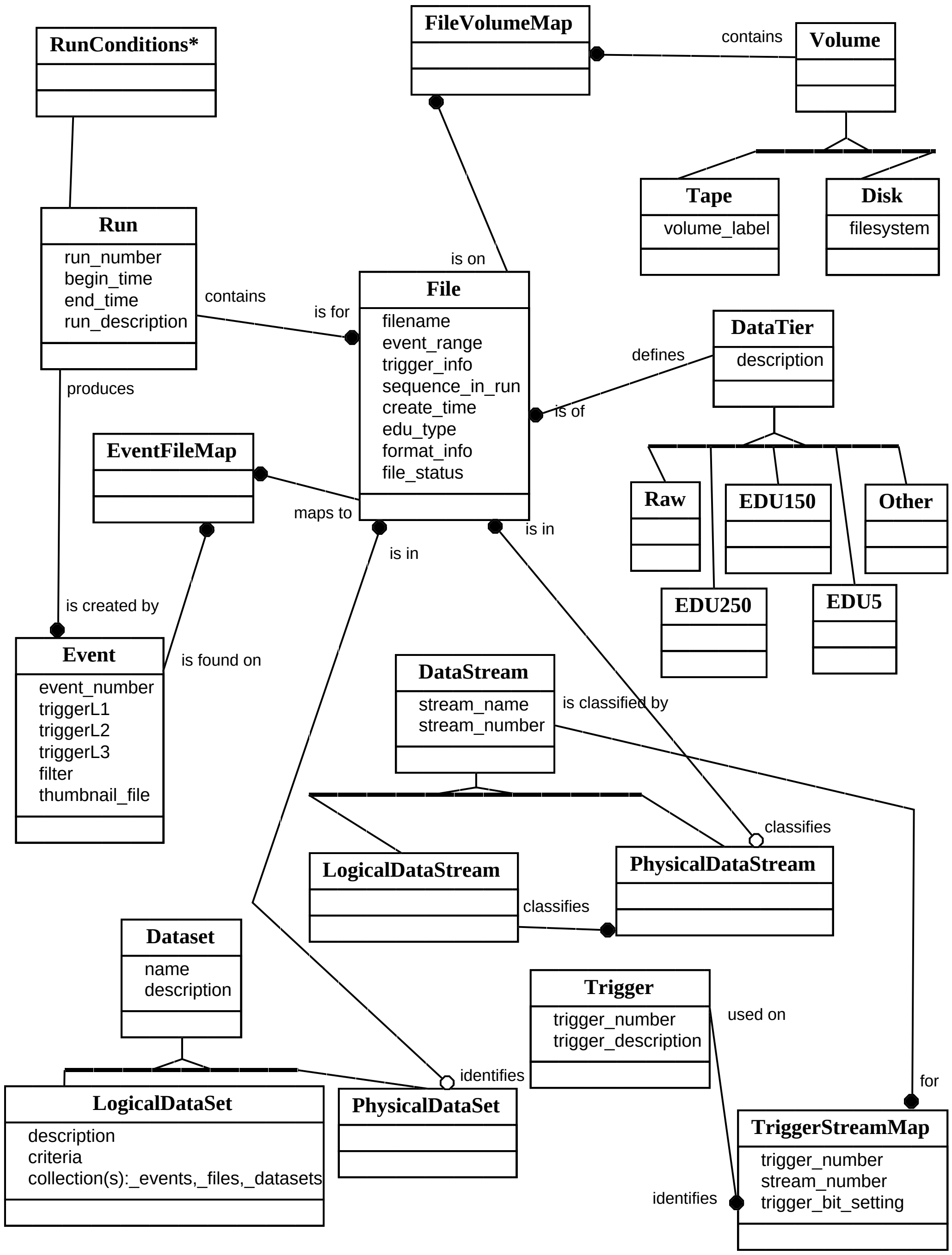


Figure 3: Possible Entity-Relationship diagram for the SAM file and event database.

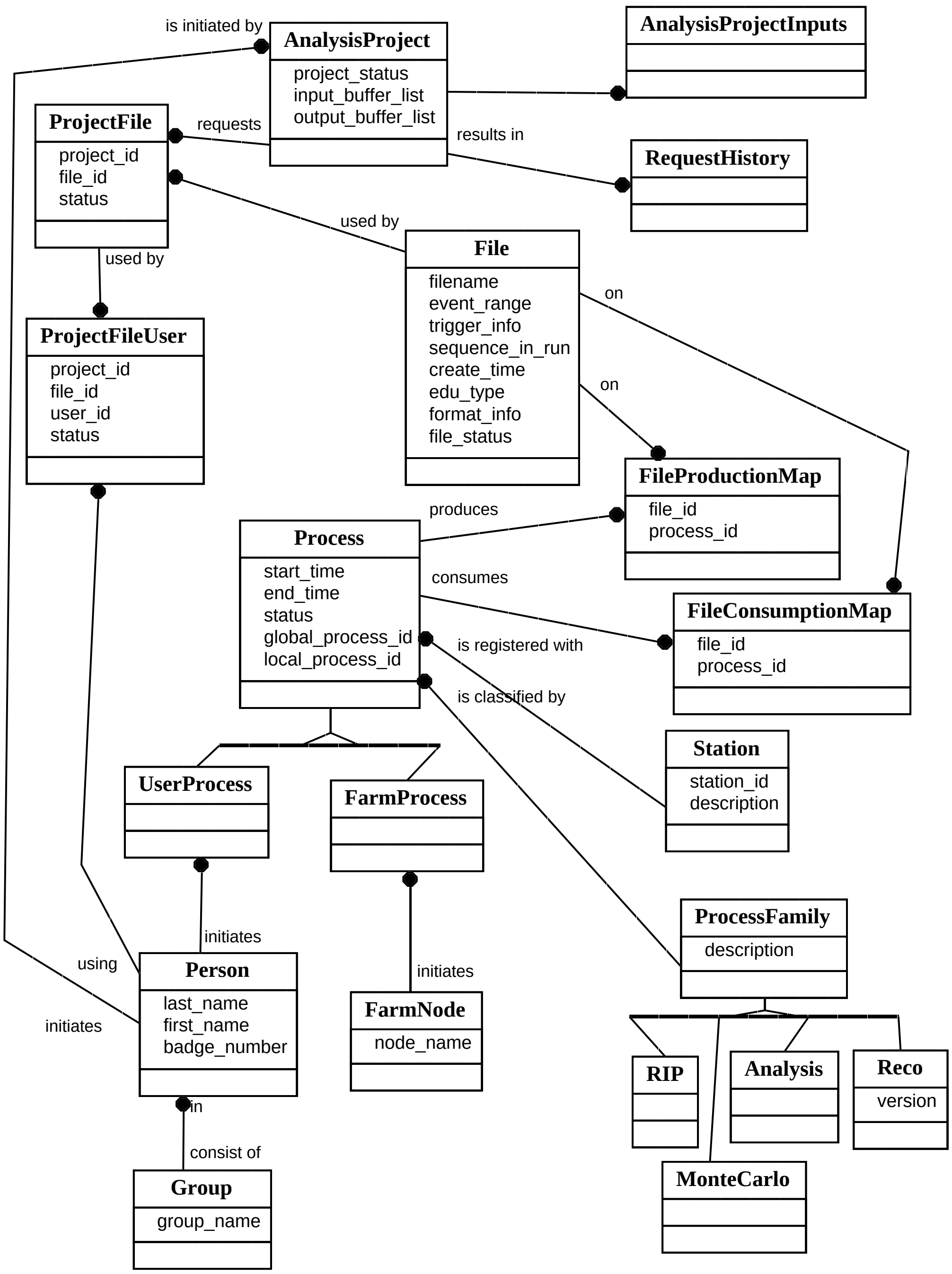


Figure 4: Possible Entity-Relationship diagram for the SAM file and event database processing accounting.

2.4.2 Data Loss Handling

While the SAM system is expected to keep data losses low, it is important that physics analyses be able to account for lost data. This places significant requirements on the tracking of files throughout the SAM system. In particular, the luminosity associated with lost files must be measured and accounted for and mappings between triggers and data streams maintained. Such needs are common to all experiments, even if the implementation differ. In order to do proper event, file and dataset book-keeping several kinds of metadata must be maintained.

2.4.3 Trigger and Stream Definitions

To understand triggers and their output to streams one needs

1. a table which contains the description of each trigger bit and
2. a table which maps trigger bit combinations to data streams

Given Run I experience, these tables will change many times over the course of the run as trigger definitions are refined. Such records probably generate only a few KBytes of metadata per day but are vital to understanding of the experiment.

2.4.4 Luminosity Data

Many physical quantities are expressed as the ratio of the process of interest to the total proton-anti-proton scattering rate. While we are taking data, we monitor the total proton-anti-proton scattering rate via the luminosity monitor. Different triggers may actually contain requirements which reduce the effective total scattering rate (ie prescales) thus we need to maintain separate luminosities for different triggers. Luminosity records need to be written out over the course of a run and a means of associating luminosity with data files is needed.

In the D0 implementation, association with data files will probably be via time stamps. 1-10 such records might be written per hour, each of around 1KB.

2.4.5 File Information

In addition to file location information, each physical data file in the system should have the following information to ensure proper normalization:

- File name
- Stream type
- Pointer to luminosity information for that file (time stamp?)
- Status of the file (good or lost)
- Reconstruction and processing information

Such file records will be generated every 1-2 minutes and be of order 1KB in size.

2.4.6 Derived Dataset Records

Any physics analysis can be thought of as a using a Dataset, which is a set of files selected on the basis of triggers and run conditions. It is useful to document and store the parameters of such datasets so that they can be reproduced later. Such datasets can be virtual, consisting only of a list of pre-existing files, or derived, where files are subject to some processing and then reentered into the SAM system.

A Dataset record might contain:

- List of all files used in that data set - can use luminosity, file and stream records to reconstruct the luminosity and validity of the set.
- Code run
- Constants used

A dataset description can be quite large with 100B/file and up to $10^5 - 10^6$ files.

2.4.7 Summary of information needed for normalization

Possible rates for data entry and quantity for information discussed are summarized from the previous sections as follows:

A rough estimate for the

Trigger information	1 per day x several KB
Trigger — Stream map	1 per day x 1 KB
Luminosity with time stamps	one every 4-20 minutes x 1KB
File records	one/file (one every 1-2 minutes) x 1KB
DataSet information	one/dataset 100B/file

2.5 The Warehouse

2.5.1 Storage Management System

A reliable Storage Management System (SMS) which manages all tape activity within Automated Tape Libraries and the movement of tapes from shelf storage into ATL is important. It is assumed that this SMS will provide access to data through file-level semantics. Additional functionality which is desirable, but not required, would enable parts of files in tape storage be accessed by specifying the byte offset and length of the required portion. With regard to putting data into the SMS, the system must also allow for placement of data into groupings of tapes by specifying file families to allow the SAM system control over physical clustering of the data. Also, some mechanism for management of the allocation and prioritization of resources within the tape storage devices, such as drives and arm time is assumed. A more detailed discussion of the requirements for an SMS to function with the SAM system are described in [5] for the anticipated DØ needs.

2.5.2 Managed Disk Cache

One of the functions desired for the system is to manage long-term disk cache areas. One example of such an area is the storage area used for the thumbnail data. This area will be significant, 10 to 15 TB of disk space which will consist of a few hundred disks. The chances of losing some of these disks is high, and thus their content needs to also be stored on serial media in the warehouse. A

Project Information			Consumer Information		
			Consumer 1	Consumer 2	Consumer n
Location	File	Status	Status	Status	Status
T1	F1				
T1	F2				
T1	F3				
T2	F4				
...	...				
Tn	Fm				

Table 1: An example of the types of information needed by the station master for Farm, Freight-train and On-demand access modes and the coordination with consumers.

convenient method of recovering from disk failure is needed and for this purpose. This recovery can be either automatic or may require someone to kick it off in the case where hardware is being repaired.

2.6 Various Modes of Access

2.6.1 File Access

The modes of access which deal exclusively in files are (1) farm, (2) freight train, and (3) on-demand files. Each of these modes has its own subtle details which require certain station functionality.

The reconstruction farm will typically deal with very large projects and process files on parallel processing hardware. At least one processing unit manager is needed to keep the system load balanced across dozens of processing nodes. There may be several concurrent activities running in various parts of the farm at any given time, for example various reconstruction versions, different physics streams, various run ranges, et cetera. Automated submission of newly taken data to a project is essential and automated monitoring and error recovery extremely important in such a large system. It is unlikely, though not inconceivable, that a file will be processed by multiple consumers simultaneously. Several error conditions must be handled automatically (1) a farm worker node crashes and the file which was assigned to it must be recovered, (2) a power failure causes all processing to stop and someone must start everything rolling again. The obvious candidate to make sure that all processing units are operating is the processing unit manager. The obvious candidate to make sure that all processing unit managers are operating is the station manager. Thus, when a project is submitted the names of scripts which fulfill the role of PUM must be supplied.

Freight train processing also may have parallel processes. The nature of this kind of data delivery is for the station buffer disk to be used by many consumers simultaneously. The number of processing units required by each consumer is determined by the speed of the processing activity. Very slow activities will require many parallel processing units in order to keep up with the rate of data flow into the system. Consumers unable to keep up with the rate will be informed that they are too slow and may be "timed-out" in order to keep the data flowing. These parallel processing units will be controlled by processing unit managers, as in farm case. Similar error recovery conditions will exist and the SAM recovery techniques will be used. Since there are several consumers processing data, if the consumer processes are not extremely reliable, the system will quickly be stopped. Therefore, the time-outs feature is needed. However, recovery for such failures may represent a significant effort if they are not corrected early in a freight train

Input files			Required meta-data						status
Family	File	MD-file	MD-1	MD-2	MD-3	MD-5	...	MD-n	ok/err
FF1	F1	M1							
FF1	F2	M1							
FF1	F3	M1							
FF2	F4	M2							
...	...	...							
FFn	Fm	Mn							

Table 2: An example of the type of information used by a station to import data in to the warehouse, say to long-term tape storage.

data cycle.

On-demand files requires user API to request broker. This is so that processing programs can be started by users which request a file or files, process them, and inform the station master what has happened. All this must occur with a few calls from the users program, and must be easy for anyone to use. A few options will be provided which provide such features as "just-in-time" processing which instructs the system to maintain two files in the input buffer simultaneously; one for processing and one to be processed next. This will make most effective use of the cache disk for many kinds of user processing. The station manager is a type of queue manager with the queue being a list of files. These files are requested from the warehouse in groups. The size of the groups are parameter tunable. The frequency depends on the size, but ultimately is determined by the status of the buffer. therefore it is driven by the rate of processing. On-demand file access requires that multiple projects be managed by a station simultaneously, and each project can be assigned a priority which will allow additional control over the system.

2.6.2 File Import

Each station's import function enables it to manage the sending of files to the warehouse for incorporation into long-term storage. The associated metadata for each file must be sent to the accountant and inserted into the event and file database. Table 2 shows an example of what information might be maintained by the import station for each import project. The only information needed by the warehouse to store away any file is the name of its file family (FF), or grouping of tapes to which the data is to be copied. In order to put a file into the accounting system, a minimum set of meta-data parameters are required in addition to the file name (F) , such as type of data, physics group, file size, et cetera. These parameters are referred to as "MD-1" to "MD-n" in the table. In addition to these basic parameters, a more extensive list of metadata is allowed which enables the importer to completely specify the files history and processing details. This information is contained in a separate file designated as "MD-file" in the table.

2.6.3 Event Picking

As was alluded to in the overview section, event picking is slightly different from the other access modes. The main goal for the system will be to get the requested events as soon as possible and notify the user when all of them are available on the PE host disk. Event picking places the burden of keeping track of event requests in the *station*. An example of the checklist which the station manager might maintain is shown in Table 3 and each entry includes the tape, file, and event id (run and event number), and a list of users asking for the event. The details of

Pick Event Project				Consumers (users)		
Event location			Status	Consumer 1	Consumer 2	Consumer n
Tape	File	Run&Event	Status	Status	Status	Status
T1	F1	R&E1				
T2	F1	R&E2				
T3	F1	R&E3				
T4	F1	R&E4				
...	...	...				
Tn	Fn	R&En				

Table 3: An example of the type of information which a station might employ to manage requests for events placed by users.

this plucking procedure are not completely worked out and may involve one of two possibilities (1) a special *picking consumer* whose sole task is to receive data from the warehouse, select out events and send them to the pick hot cache, or (2) a feature of the warehouse which interprets the request for an event as file, byte offset, and byte length and sends, just this piece of data, directly to the pick hot cache. Since requests for large numbers of events may require a long time to fulfill, requesters will be sent periodic updates on the status, and allowing them to get available events. Access to these events ill be through some convenient mechanism, possibly a script or other approach, available to the user. When the user gets the event requested, his entry in the consumer table is marked as complete. When all consumers have been check off, or after a timeout period, the files in the hot PE cache will be moved to the warm PE cache by concatenating them together in logical groups and the resulting files stored to tape.

There are two competing objectives for this mode of access: (1) satisfy requests quickly , and (2) pool requests as long as possible to reduce the total number of tape mounts. In an attempt to meet both, it is likely that a few "levels" of picking station might be established which pool over various time periods. The request manager sending pick requests to a station manager would, therefore use the size of the request, in terms of the number of tape mounts required, as a criteria. There would likely be an hourly, daily and weekly station. Single event, or a few event, requests would go to the hourly station and be selected right away. Larger jobs would go to the daily station and very large jobs would be pooled and started once a week. Alternatively, a system of priority event picking based on the size of the request may provide the needed pooling without actually pooling requests over fixed time intervals.

Another variation on the weekly picking might involve allowing only events in a certain run "window" to be selected. This window might be the weeks worth of data which had been reconstructed two weeks ago, for example. The logic being that this allows a week to analyze of the thumbnail data and make event lists for picking. This type of event selection may be required to make it possible, without swamping all tape resources, to go through lots of data.

Following is a list of several of the features which are being planned within the SAM event picking system.

1. **Bookkeeping:** The PE system must have very good bookkeeping system for all the requests and current status of events. When the user creates PE request the list of wanted events is given. The PE system will search for all the events available on the PE hot first, then on the PE warm and rest of the events will have to be accessed from via warehouse. In case the event can't be read the user will be notified.

2. **Priority system:** Each request is assigned a priority level based on the number of events

and possibly the number of tapes required to be mounted to get the events (possibly the size of all requested events too). This guarantees that the small requests with only a few events will be satisfied quickly.

3. **Queue system for the warehouse:** The PE system should have the queuing system for the tape access. Assuming there will be about one entry for each tape that needs to be mounted for the PE system with the list of the events to read. The queue will be ordered by the priority and the time waiting of the request with the highest priority.

4. **Size quotas:** To ensure that the system will run smoothly with the limited disk space for the events to be accumulated there will be quotas for the size used by each request, user and group.

5. **Backup system:** All picked events will be merge into reasonable size files and stored on the tape as a PE warm. This provides efficient and quick access to the most interesting events.

6. **PE hot cleaning system:** The events accumulated on the PE hot and backed up on the PE warm will be deleted when the space is needed or there are no more requests waiting for them.

2.7 How The System Might Function

There are many additional details of how the system may function which need to be covered. In order to perform this, a few specific examples of interaction with the system are provided and how the various activities might be managed.

1. **User and Administrative Interaction:** The SAM system allows different levels of interaction with the system. The main difference between the actions allowed is that administrators are may create very large projects, while general users are limited to small projects. Users, of course, have other limitations when they are registering to a preexisting project created by someone else. Following are examples of type of actions a user or administrator might take.

   (a) Test/create a project by submitting a request.
   (b) Launch a project - The project is ready to start.
   (c) Edit a project - Add files/event to or remove them from the project.
   (d) Start a project.
   (e) Stop a project.
   (f) Cancel a project.
   (g) Start/stop consumers registered to a project.
   (h) Add/remove consumers registered to a project.
   (i) Check status of project.

2. **Defining Projects:** Work is submitted as ”projects” or ”orders”. These Projects are comprised of the files needed to be delivered to satisfy a particular request. A request may have several forms, but generally defines a logical set of data based on certain constraints. These constraints include the following: run number, run range, event number, number of

events L1 trigger, L2 trigger, L3 trigger, cluster ID, reco version, EDU type, integrated luminosity. There may be other constraints, the only requirement is that the constraining parameter is in the event and file metadata.

3. **Launching a Project:** A user or administrator launches a project with the following steps: Send a request (constraint or file list form) to the *Request Broker* (RB). The RB consults with the *Translator* to estimate the size of the request in terms of files. By consulting with the *warehouse* inventory, an estimate is made how many tape mounts will be required to satisfy the request. The request broker checks the user's authorization level. If the user is not authorized the level of work needed to fulfill the task, a message is returned explaining this. If the authorization is satisfactory, an estimate is returned with a summary of the size of the job (# files, tape mounts, Est. Time to complete, and so forth) and the requester is asked if the project should be created. If the project is continued, additional info is needed to completely define the project, including the destination for output so it can be sent to the warehouse on completion. The request in *project* form is sent to the Station Master. The Station Master establishes a table which has one line for each file. Each line has the file name and a status word which will indicate the state of that file, code. The project is launched by the requesting user or admin person issuing a start-project command.

4. **Registering to a project:** To connect to a project a user looks at a list of currently running projects and selects one. He then requests a token from the Station Master for this project. When the Station Master grants a token to the user, he establishes an additional column in the projects status table so the processing of this user can be monitored with entries like start_time, completion_time, and error. The Station Master does not care about additional details about the users processing. If a project is started and there are no active token-holding consumers, the project will go into an idle state.

5. **Processing Units:** The part of the system which processes delivered data is a Processing Unit (PU). Each user can have one PU, or many PUs working on the data in the station on the input buffer. If there are many PUs, they can all run on a single machine, or be running on many independent CPUs. It is a separate task to make sure that these PUs are all running, and they should not interfere with each other, therefore the SM must communicate to only one at a time. To work on a file for a given project, the PU simply shows his token to the station master to connect to the project data stream. The SM notes the starting time and waits for the Processing Unit to send a completion or error signal. There can be many Processing Units working on the data in the station (on the input buffer) from different users, or the same user processing in parallel. When all PUs with tokens for the project are finished with a file on the IB, within a certain time period (to be determined by the processing duration distribution of token-holding users), the SM will send a message to the IBM to remove the file and the warehouse will send more. The PU uses two buffer managers, the input buffer is from where the input comes and the output is to where the output goes.

6. **Processing Unit Manager:** The PUM communicates with the station manager to determine how many files are left unprocessed for a particular token. The PUM then starts PUs and gives them tokens. The PUs present these tokens to the station manager when they connect to the project stream. The task of the PUM is to make sure that all PUs are busy and that the load is balanced, in the case where there are many nodes, such as on a farm. The PUM could be dispatched by the SM, or could be controlled by a user. For production activities, it would be strongly preferred for this to be under the control of

the station master, in case there were some malfunction which caused the PUM to stop. Therefore, when a user or administrator connects to a project, he would provide information about the PUM as part of the request.

3 System Hardware Architecture

3.1 Objectives

As we build the hardware architecture, the goals are to provide a system which (1) satisfies the diverse needs of each experiment, (2) efficiently utilizes the hardware resources, (3) is easy to maintain and troubleshoot, and (4) allows for new hardware to be added without destroying the model. There are four general categories of hardware included (1) serial media data storage, (2) random access data storage, (3) network, and (4) CPU. Serial media data storage includes all hardware related to the storage and retrieval of data on serial, i.e. tape, media. This includes tapes, tape shelves, Automated Tape Libraries (ATL), and the tape drives needed to read and write the serial media. The second category of storage, random access, refers to disk storage and there are three categories of this (1) buffers used for bandwidth matching between two parts of the system, (2) caching disk used to keep data short-term which may be needed by several users, and (3) long-term stores which are maintained for very long periods of time - weeks or months. Networks include three general types (1) high bandwidth switching networks, (2) local area networks used on site, and (3) the wide area network used for transferring data to and from remote sites. Finally, CPU is used in the system in three capacities (1) I/O to and from storage or buffer devices, e.g. mover nodes for tape storage or I/O nodes for buffer disk areas, (2) database serving, and (3) data processing CPU.

3.2 Database Server System

A separate, dedicated database server system should be provided to support the various meta-data and other databases needed for SAM. By centralizing the database server, hardware and software licensing costs can be reduced. It is expected that most, but not necessarily all, other SAM systems would be equipped with the database client licenses and software to allow remote access to the server.

A modest sized Enterprise-class SMP system would likely serve to power the database server. The system would be equipped with locally attached disk and tape drives for backup. As the database server is critical to the experiment, extra funds should be used to provide a highly-available system. This may involve redundant systems with fail-over, RAID disk arrays, etc. to insure maximum availability of the system. The extra expense for these features, concentrated in this limited part of the overall SAM system will have a big pay-back in operational efficiency of the complete complex.

3.3 The Data Warehouse Components

The data warehouse includes any hardware which is used for long-term data storage. This includes Automated Tape Libraries, Shelf tape, and long term disk storage. The data stored in these areas include (1) Picked events stored on disk cache (PE hot), (2) Picked events stored on tape (PE warm), (3) Analysis disk cache (slush disk), and (4) Long-term data stored to tape. Some parts, such as the analysis disk cache, might appear to SAM as logical parts of the warehouse though they might be physically attached to hardware more closely related to processing CPU. Import staging disk is used to import data into the SMS which controls the robot and shelf tape store.

Application	CPU	Bandwidth	concurrent users
Monte Carlo	high	low	one
Farm Reconstruction	high	high	one
Freight train processing	medium	high	few
Analysis-general	low	high	many
Analysis-thumbnail	low	high	many
Event Picking	very low	low/high	many
Import/Export- Network based	very low	low	few
Import/Export- Tape based	very low	medium	one

Table 4: Characteristics of various data processing applications.

3.4 Example of Various Hardware Configurations

Several example hardware configurations are illustrated in Figures 5, 6, and 7. The various arrangements are characterized by the throughput and nature of the network connections, the nature of the processing CPU(s) employed, and the size and locality of the buffer/cache disk(s). Table 4 summarizes qualitatively the relative amounts of CPU, bandwidth, buffer, and concurrent users required for different processing areas.

- **Processing Farm** (Figure 5a): The processing farm has high demands on CPU and bandwidth. The most likely configuration is a large number of commodity CPU processors (workers) connected to a non-blocking high-speed network switch. An IO server is connected to the switch and receives input data from tape storage ATL and collects output data from the worker nodes. The network connection from the ATL to the I/O node, and the disk buffer local to it must support very high throughput. Only minimal disk storage is needed at each worker and modest bandwidth through the switch is needed form the I/O node. An SMS controlled import buffer is probably needed to transfer farm output data to SMS controlled media.

- **Data Mining Facility** (Figure 5b): Here we consider mining data from a disk resident data set, such as the thumbnail. The data would be transferred, whenever updated, from the ATL through an I/O node, and distributed evenly to cache buffers local to commodity CPU mining nodes. Requests to sift through the data are sent from a coordinating process and submitted in parallel to each of the mining nodes. The results of a search are sent back to the coordinator and glued together. Because the data is distributed and cached among the mining nodes, substantial disk resources are needed on each mining node.

- **Analysis Facility** (Figure 6a): For general analysis, one or a few large SMP machines may be useful. Such processing units would receive data from the ATL over high throughput networks. They would have substantial amounts of user disk buffer, and a large cache buffer disk area. The user disks would be "owned" by physics groups or individual users and it would be their responsibility to manage and back them up. The cache buffer would be managed as part of the central store, and its management would be part of the SAM responsibility. An import buffer, internal to the warehouse, is needed to store analysis output data to ATL controlled media.

- **Event Picking Facility** (Figure 6b): For event picking from tape the goal is to do the actual event selection from the data files as close to the drive as possible. The network throughput from the SMS to the Pick Event processor will be small if the event filtering is performed in the SMS mover node. On the other hand, if the events are not extracted from the files until the data is on the work buffer disk, then the bandwidths are significantly higher. Selected events are cached in a large buffer area which is managed as part of the warehouse, though physically it is connected to either the pick events node or an analysis facility.

- **Import/Export Network Facility** (Figure 7a): For off-site users, a facility is required to have data delivered from the ATL (export) to their local institutions and to send data from their homes to the warehouse (import). Consumers would connect to an I/O system and transfer data through the Wide Area Network. The I/O system would have enough buffer disk to meet the user demand for this system, but would be at least large enough to stage several hours of transfer, an entire tape for example. An import buffer, internal to the warehouse, is needed to store import data to ATL controlled media.

- **Import/Export Tape Facility** (Figure 7b): For moving very large quantities of data to and from outside institutions, a tape facility is required. The operation is straightforward, data is requested from the warehouse and delivered to the I/O buffer. From the buffer it is spooled to tape. An import buffer, closely connected to the ATL and managed by the SMS , is needed to store import data to ATL controlled media. If the media being imported is the SAM as that used in the ATL, and the format of the data can be imported directly to the SMS, it would be preferable to simply import the metadata and add the tapes to the SMS facility.

3.5 SMP/Enterprise Server

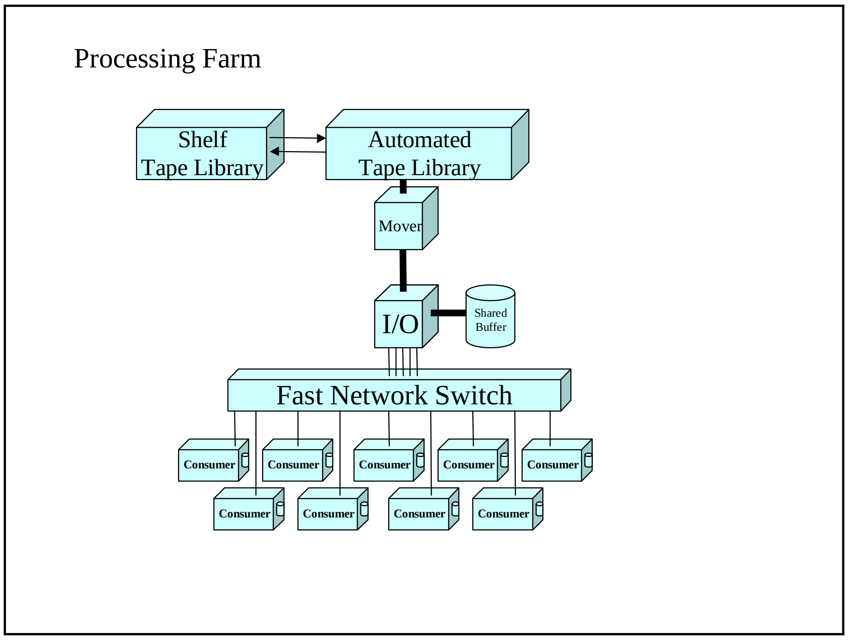
This system would primarily be used for post-reconstruction analysis and production. The emphasis here is primarily on the ability to provide high IO rates mostly to local disk. This monolithic system approach also brings with it the benefits of having a single system to manage and the problems of sharing resources among disparate user communities. A possible downside to a single system configuration is a limit on IO connectivity (i.e., how many local disks can be connected to the system).

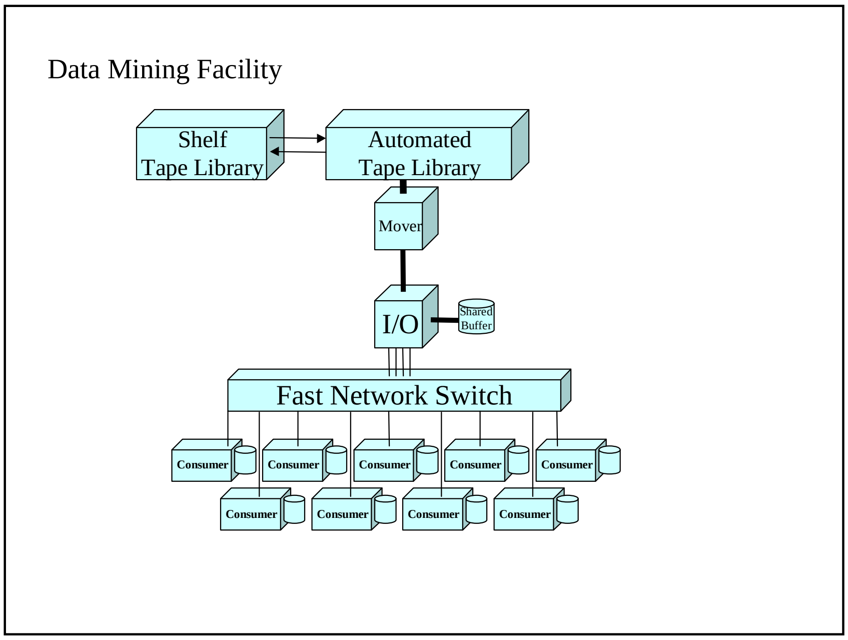
However, it seems unlikely at this time that Fermilab can afford a single Enterprise system capable of handling the Run II analysis computing needs or indeed if such a single system even exists. Instead, we will have to use a few large Enterprise-class systems. This complicates the system management but might alleviate problems with resource sharing and IO connectivity. One possibility opened is to segregate the user community by physics group among these systems.

These systems will need to have local disks for system software, temporary an scratch areas and spool buffers for SAM. If the user community is to be segmented, then their permanent file could also be on locally attached disk. Otherwise, some sort of central file server (such as a dedicated NFS-based file server) to allow uniform access to these files would be preferable to creating a web of NFS cross-mounts.

3.6 Distributed Network CPU

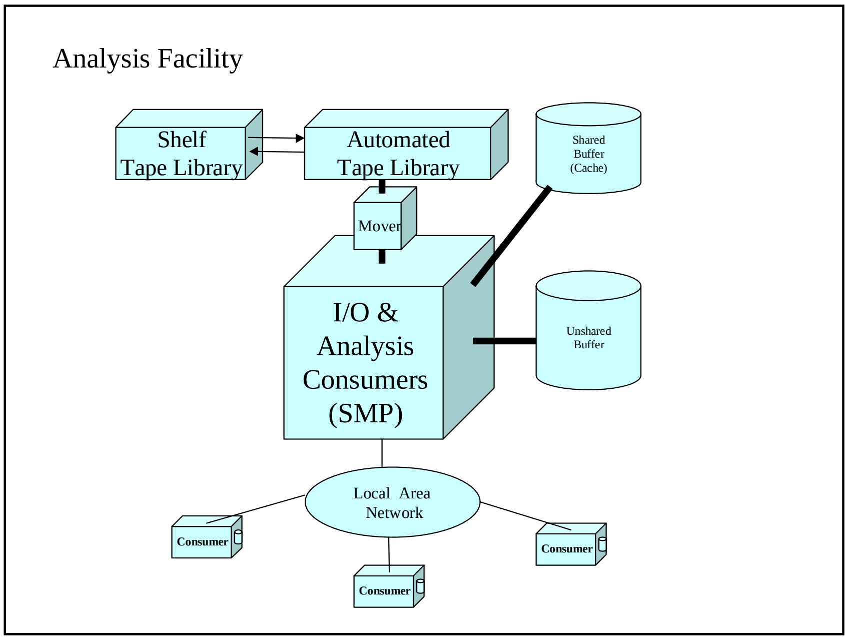
Classically this has been the form of Farms for reconstruction production computing at Fermilab. In its new incarnation, the worker systems (typically small single CPU or small SMP systems) are connected by a switched network architecture with 100BaseT being the likely connection method. Typically the high CPU to IO ratio of reconstruction favors small systems and needs only modest network bandwidths.

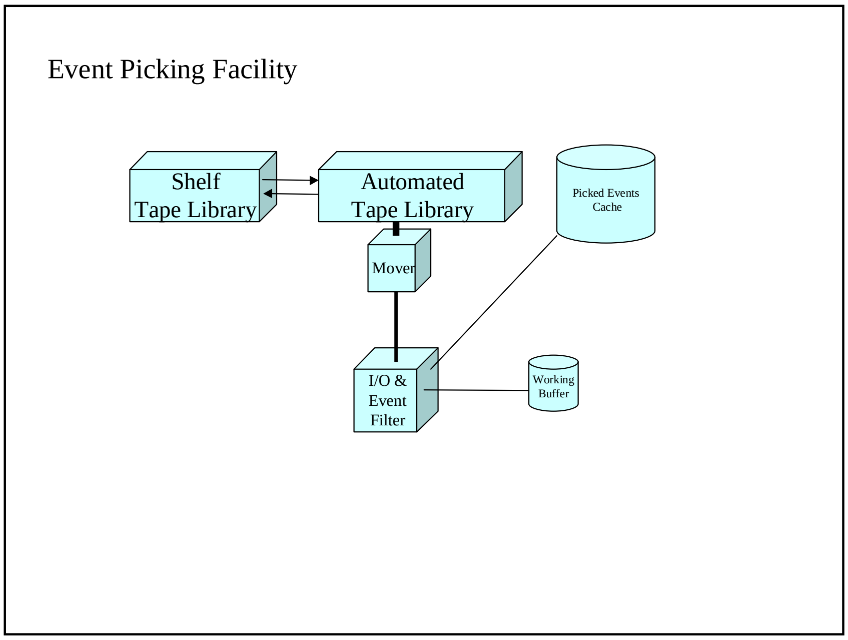




1

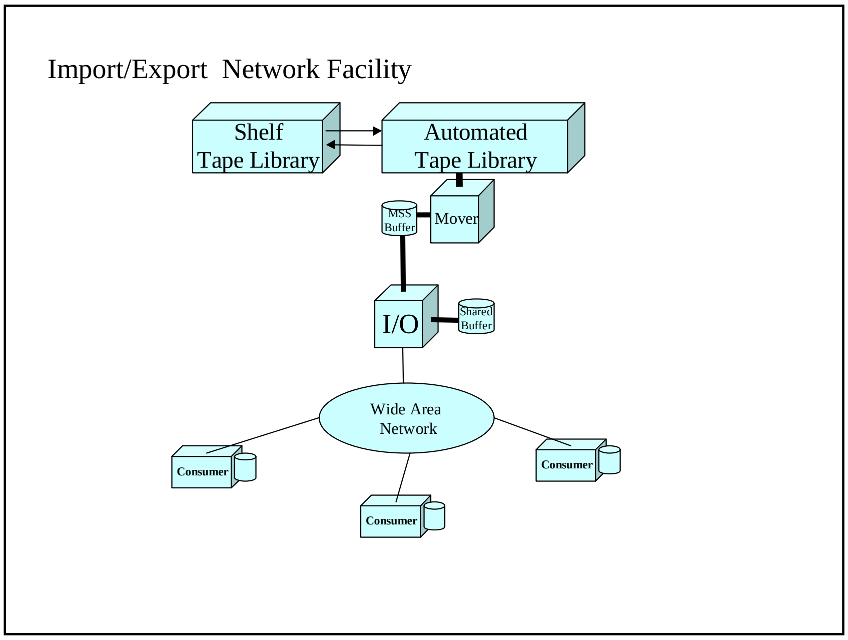
Figure 5: Examples of hardware configurations for the sam architecture.

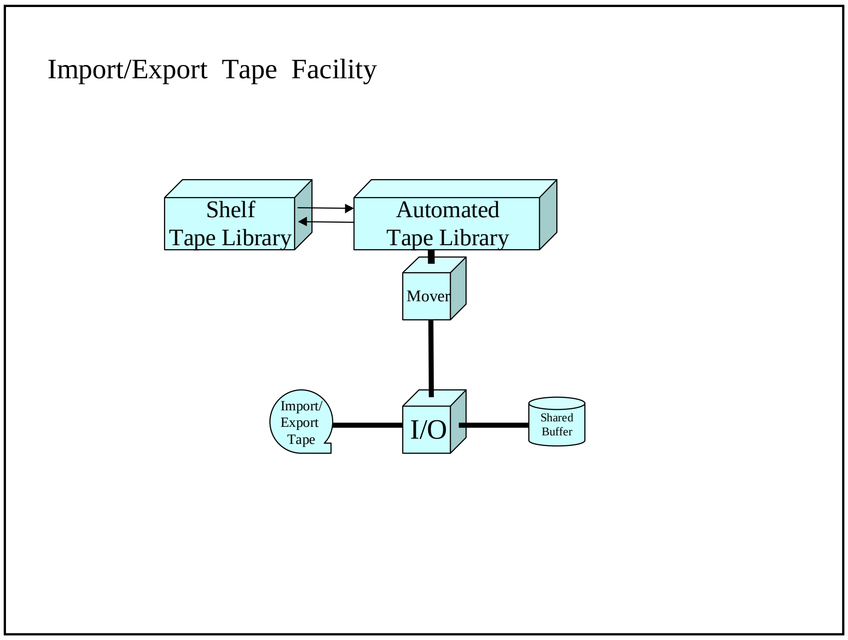




1

Figure 6: Examples of hardware configurations for the sam architecture.





1

Figure 7: Examples of hardware configurations for the sam architecture.

An IO server or controller system sits between the worker nodes and the Data Warehouse. The IO server has a high speed network connection to the robots and high speed disk arrays to be able to handle data at the tape drive IO rates allowing IO transfer rates to the worker nodes to be lower and more cost-effective. The IO server node can also act as the central control point for the Farm, managing the data access and job execution on the worker nodes.

While Farms have worked well for reconstruction computing which has a high CPU to IO ratio, it is unclear if a similar arrangement will work for analysis computing with its higher IO needs. One possibility is a group of small systems acting as a batch computing farm in conjunction to a large SMP analysis computing engine.

3.7 Off-site Data Delivery

Data can be moved off-site via the network. This can be done to either make local copies of the files to be accessed by a remote system using SAM. Off-site network bandwidth limits and the desire to use the robot tape IO bandwidth efficiently (or maybe that's maximally) suggest that off-site network access to the Data Warehouse be buffered by an IO server system with local high speed disk. This system can accept the data a full tape speed and buffer it for slower off-site transfer or access.

The other possibility is for bulk delivery by making tape copies at Fermilab to be shipped off-site. This does not necessarily mean the use of the same media as will be in the Data Warehouse robot. An export server system with local buffer disk and tape drives would prepare the tape copies. In particular this was envisioned as a mode of operation of the new Tape Copy Facility currently under development.

4 Summary

The system software and hardware architectures for the Sequential Access Model have been designed to meet the requirements for the Run II data Access. The system intended to be flexible enough to meet many access modes and various hardware configurations. It is also modular and scalable, so changes and additions can be made to modify and/or scale up the system for future needs. There has been an attempt to keep the system simple to use and easy to maintain as well. Thought has been given to reliability and error recovery in the system, so that it is easy to keep running and straightforward to recover from failures. Several hardware configurations have been considered to understand how the system might function, and the various needs for storage, network, and CPU resources.

A SAM Control and Data Flow

Figures 8 and 9 reflect our current understanding of the SAM data flow. They outline the flow of both the control messages and the actual data (event files). Both figures omit accounting and auditing because 1) every action is understood to produce accounting information, thus, explicit showing would only clutter the figures, and 2) the methods of monitoring will change dynamically.

In both figures, the ellipses denote processes (either stand-alone executables or callable routines), cylinders denote disks (either cache or buffers), thick arrows show the flow of the main data (files containing events), thin arrows show flow of control messages and auxiliary data (such as the database data about the vent data) exchange, and the pairs of thick horizontal lines denote the auxiliary data. (Its proximity to other components indicates intended usage rather than physical location; virtually all such data can potentially be stored in one database.) Shade contours indicate cardinality more than one.

We envision the following components. There may be many users (which are analysis processes, farms, etc) and many stations. There is conceptually single database and single store manager.

Figure 8 addresses the Freight Train, the Files on Demand, and, to large extent, the Farms access modes. To initiate access, user's application creates a project by sending a request for files (in the form of a logical dataset or explicit file list) to the Station Manager (SM) of any capable station. The SM returns a time estimate for the project, based primarily on the number of tape mounts that will occur during the project. The latter is computed based on the files' location (obtained from the Database) and the current state of the tape mount queue (obtained from the Tape Manager part of the Store Manager). The part of the station that provides interface to the DB is loosely called Translator. The Store Manager is described later. Upon the user's approval the project is defined at the station in the form of (essentially) a file list; the Station Manager launches a Project Manager.

A user's application can join, quit, suspend, resume, cancel, etc. a project by contacting the Project Manager of the Project. The Project Manager maintains persistent project data, such as the complete list of files and the checklist of user-file statuses. The Project Manager updates this information whenever a user interacts with it. In addition, the PM allocates a Process ID (PID, not to be confused with operating system pid's) for the user application, which uniquely and persistently identifies the user application (i.e., the producer/consumer process). The PID is used to track users' progress in the project and to track files' creators.

A station has many (at least the number of physical file systems mounted) Input Buffers implemented as disks. Although these disks indeed primarily serve as *buffers* (especially for FT and Farms modes), together they form a distributed disk *cache* for files. There is one Input Buffer Manager per each buffer, project or station. An Input Buffer Manager uses the project data to ensure that each active process (or family of processes as in the case of Farms) consumes every file; it also serves the distributed disk cache by communicating files from/to other Input Buffer Managers. In addition, an Input Buffer Manager may mediate the file download from the Mass Store.

The part of the user application that communicates with the Input Buffer Manager(s) is called the Input Data Stream Manager. At minimum, the Data Stream managers provide an intelligent data stream to the application and sends signals for opening, closing, buffer-locking, etc. of files to the Input Buffer Manager(s).

The Tape Manager in the Store Manager optimizes (globally) the tape mounts. It maintains a persistent data structure such as a queue of (File, Tape) pairs representing the tape-mounting work to be done, as well well a list of destinations (essentially Input Buffers on stations) for each

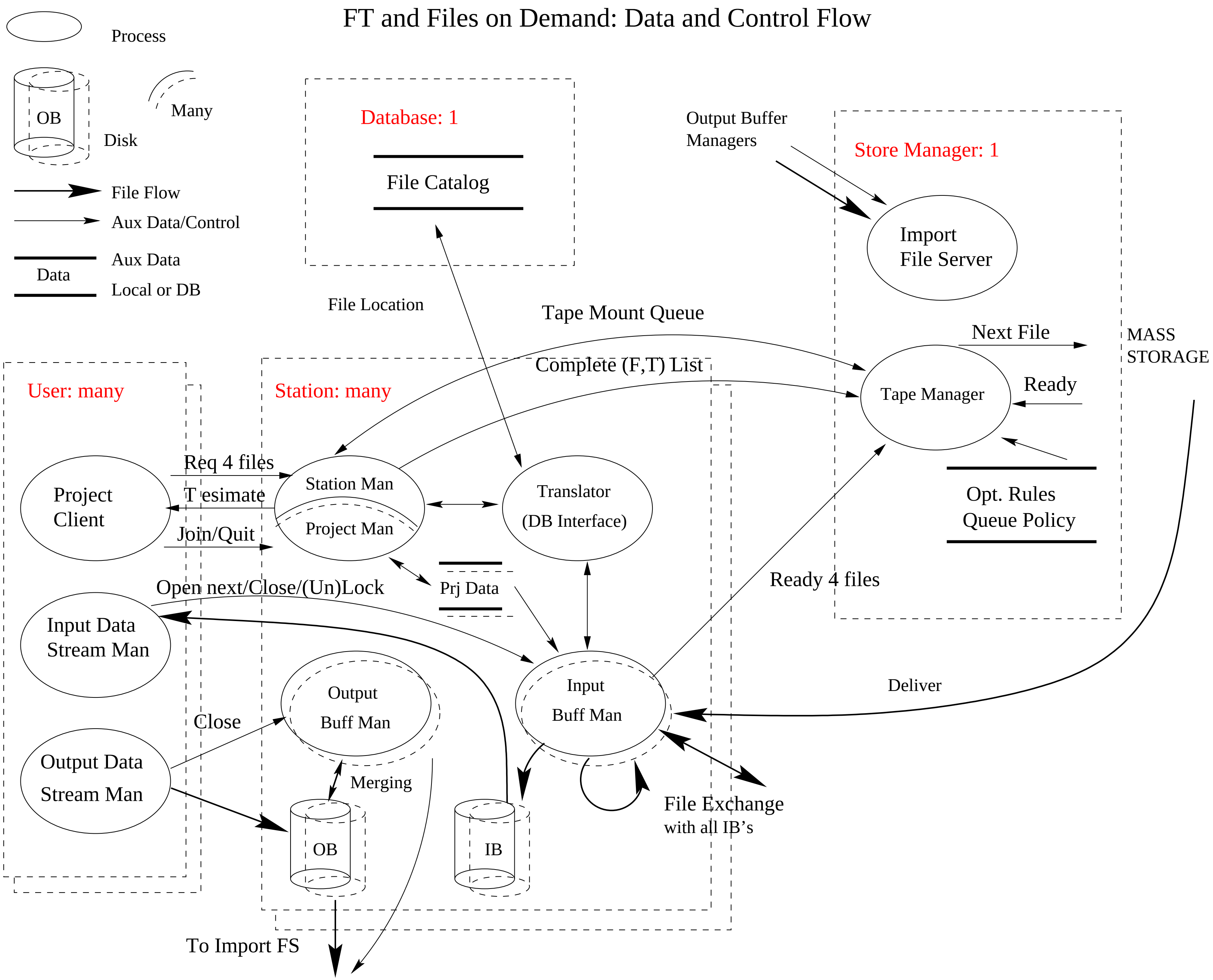


Figure 8: Data flow diagram for SAM file access including Freight train, Farm and File on-demand modes.

file. It requests tape mounts from the Mass Storage as resources become available. The Tape Manager uses information such as queue policy and rules for optimizing the tape mounts while ensuring the unbiased sampling of physics streams. This data can be modified only by a SAM administrator.

The import side of the access modes involved is shown in a lesser detail. Users' Output Data Stream Managers, When an application closes an output file, its Output Data Stream Managers signals the Output Buffer Manager which merges, if necessary, the file with other output files and schedules the file import to the Mass Storage with the Import File Server at the Store Manager.

Figure 9 shows The Event Picking mode, which is largely a mere simplification of the above file modes. Similar to the above, an application requests a logical dataset for events. The necessary tape mounts, if any, are scheduled with the Tape Manager of the Store Manager, perhaps with a low priority and/or a pooling. The Mass Storage delivers files containing the selected events to a small, not much greater than a single file size, buffer belonging to the Store Manager. The Pick-Event Cache Manager uses experiment-specific rules to extract the selected events from the delivered file, perhaps envelope into a special file, and store the events on disk, i.e., in PE Cache. The requester (human user) is notified of the delivery. The user then uses an Input Buffer of any station to read the events from the possibly remote PE disk or (which is discouraged) replicate the events in a private store area.

References

[1] RIP document

[2] "Sequential File Access Model Evaluation and Design for Run II", October, 1997.

[3] Lee Lueking, "Flows and Controls in the Seasons of SAM"

[4] "Requirements for Sequential Access Model", June 1998

[5] DØ Run II Computing and Planning Board, "MSS Software for Run II - DØ Evaluation", June, 1998.

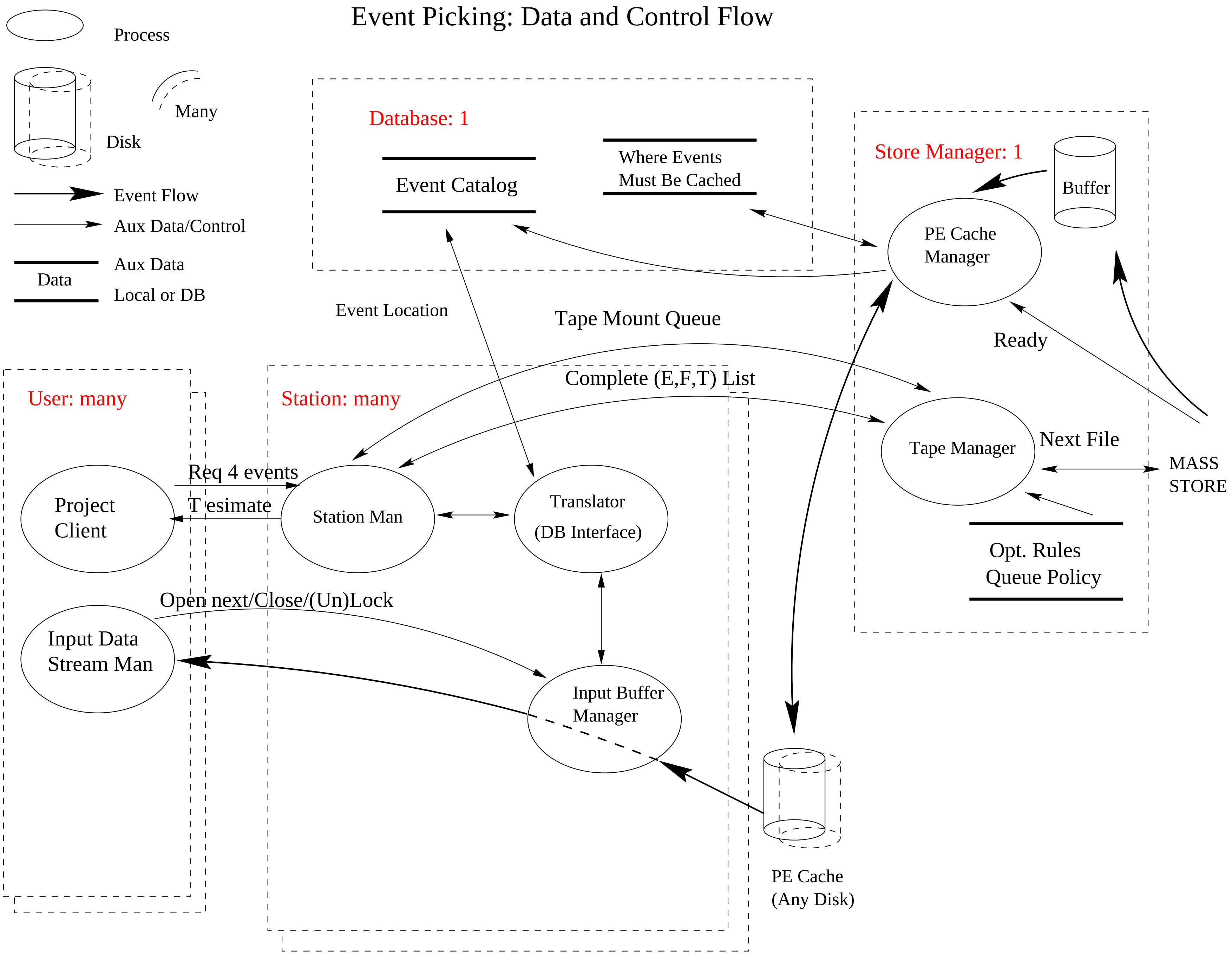


Figure 9: Data flow diagram for SAM pick event file mode.