The IEEE Mass Storage System Reference Model

J. Shiers
CERN, Geneva, Switzerland

Abstract

The IEEE Reference Model for Mass Storage Systems provides a basis for the development of standards for storage systems. The model identifies the high level abstractions that underlie modern storage systems. The model itself does not attempt to provide implementation specifications. Its main purpose is to permit the development of individual standards within a common framework.
High Energy Physics has consistently been on the leading edge of technology and Mass Storage is no exception. This paper describes the IEEE MSS Reference model in the HEP context and examines how it could be used to help solve the data management problems of HEP.

1 Introduction

These lectures cover the evolution of the IEEE reference model for mass storage systems and examine the suitability of the model for High Energy Physics.

The first lecture provides an overview of data handling in High Energy Physics, followed by a description of a software solution that was developed at CERN to meet the needs of the LEP experiments. Finally, the requirements of future experiments, such as those planned for the Large Hadron Collider (LHC) at CERN, and the Superconducting Super Collider (SSC) in the US, are discussed.

The second lecture reviews existing storage systems, with particular emphasis on those that influenced the IEEE MSS model and those that have been developed along the guidelines of the model. Included in this lecture is a discussion of network file systems. Although such systems were typically developed independently of the model, they are sufficiently important to warrant inclusion.

The final lecture is devoted to the model itself and how it can be used to build a global solution to the problems of data management in HEP. It covers the history of the model, the current status and the possible impact of the model upon HEP. Areas that the model does not cover are highlighted and possible solutions to these problems described.

Non-goals are a discussion of storage hardware, the data model or languages. Although these will be touched on in passing, they are not the main theme of the lectures.

Given the title of the lectures, it is perhaps useful to explain why topics other than the IEEE MSS model itself are discussed. There are a number of important reasons for this, as described below. Firstly, the model itself is still evolving and it is useful to examine its origins and explore future directions in which it may develop. Secondly, many of the systems that will be discussed had strong influence on the development of the model. Indeed, it was the belief of the developers of the model that a credible standard could not be achieved without demonstrable prototypes. In addition, it is important to understand how the model could be used to solve the particular problems in High Energy Physics.

Virtually all of the information that is contained in these lectures has been presented at an IEEE Mass Storage Symposium.

2 Data handling in High Energy Physics

2.1 Introduction

My own interest in data management started soon after starting my physics PhD at the University of Liverpool. The High Energy Physics group had a small IBM 360 computer that ran the MFT operating system. I was immediately faced with statements like the following:

```
//G.FT11F001 DD DSN=GEOHYB.P4C79.FILM47,DISP=NEW,
// VOL=SER=939727,DCB=(RECFM=VBS,BLKSIZE=32756),
// UNIT=6250
```

This particular statement contains a simple error, namely that the `/LABEL` qualifier is missing. As a result, the first file on the volume will be overwritten. Such mistakes do not increase one's popularity with fellow students, particularly when they have spent many hours of computer time to write the files destroyed by this omission.

Like many other experiments at the time, our *tape catalogue* consisted of a sheet of computer paper on which we were supposed to mark those tapes that we had used. Surprisingly, some groups still use electronic sheets of paper - a unstructured file that they modify with a text editor.

The above JCL statement is somewhat atypical, in that a semi-meaningful dataset name is used. This is still uncommon practice, even today. Many experiments prefer to use the same dataset name for all files.

Another lesson in the need for data management came shortly afterwards, when using the so-called `PUBLIC EXEC` DISKTAPE. As the name suggests, this exec file could be used to copy files from disk to tape. Unfortunately, it did not make use of one of the more esoteric features of IBM's *job control language*, namely the `UNIT=AFF` qualifier. This allowed one to specify that one would like to use the same physical drive as was used for a specified stream in a previous job step. Failure to specify this absurd level of detail meant that the system rewound and unloaded the tape between each step and then requested a new mount, almost invariably on a different unit. This, naturally, did not increase one's popularity with the tape operators.

After these and many similar experiences, it was clear to me that some form of intelligent software was needed. Whilst working on the NA9 experiment at the Max Planck Institute in Munich, I developed a package that provided access to data by name. The system automatically generated the necessary JCL statements, even managing obscurities such as `UNIT=AFF` etc. The system knew the correct record attributes of the different data types, and generated appropriate `DCB` statements on output accordingly.

Surprisingly, to me at least, this system was not popular. People preferred to hard-code specific tape numbers into their jobs. I used to liken these people to those who still kept (and probably still keep) dusty punched card decks in their desk drawers. However, I had the last word as I would occasionally repack volumes (automatically, of course) and so their jobs with hard-coded tape numbers would fail.

2.2 Data processing

Data processing in a High Energy Physics experiment involves a number of steps:

- Raw data is acquired (or simulated)
- The raw data is processed through the reconstruction program, producing Data Summary Tapes (DSTs).
- The DSTs are then reduced for further analysis, typically by reduction into Ntuples for processing by PAW.

It is interesting to note the widespread use of farms in the data processing of a modern HEP experiment. The simulated data is typically generated on a dedicated farm, such as the Central Simulation Facility (CSF) at CERN. The reconstruction is generally done at an experiment specific farm close to the detector itself. The data reduction is increasingly performed on dedicated farms such as Shift. Finally, we see the use of farms such as PIAF for parallel visualisation. This strong move towards distributed computing has a major impact on the whole question of storage and data access. This shift is often paralleled to the move from the Ptolemaic view of the Universe, where the earth (or CERNVM) was considered the center of all things to the Copernican view. [1] As we shall see later, one of the main problems with the storage systems developed in the 1980's was that they were based on a centralised model and do not adapt well to the distributed environment.

Until recently, only one level of offline storage has been used. This has changed slightly with the introduction of tape robots, where robotically mounted volumes are placed higher in the hierarchy than manually mounted volumes. It is clear that we will use multiple types of media in the future and it is important that the data access characteristics of the different types of data are taken into account. Raw data, for example, is processed infrequently - ideally only once - and sequentially. DSTs are processed much more frequently - at least once a day in the OPAL Shift configuration. Here random access is essential. Finally, Ntuples are processed many times per hour and should, for efficiency, be mapped to memory. [1)]

2.3 Data volumes and rates

The CERN tape management system currently controls over half a million volumes. Assuming that all of these volumes are 3480s containing 200MB of data, this corresponds to some 100TB. Approximately 2% of the volumes are moved in or out of the central tape vault per week with a tape being mounted every 45 seconds on average. Over 300 thousand commands are executed by the tape management system per week.

Raw data sizes are currently in the range 100-500 KB/event. After reconstruction, the size of an event tends to grow by between 10-20%. This is because the raw data is normally carried through to the output stream to facilitate reprocessing.

2.4 Event directories

A relatively new technique that is used by two of the LEP experiments is the concept of *event directories* [2]. This technique, developed independently at DESY and on the OPAL experiment at CERN, uses a list of events with their characteristics, file, record number and offset. Rather than make multiple streams from a common set of DSTs, one stream for each analysis group, event directories permit a single set of files to be shared by all groups and users. It is clear that such techniques will be increasingly important in the future, if one is to reduce the overall disk space requirements.

2.5 Data import and export

Data import is primarily concerned with simulated events which are generated outside of CERN. These events are often generated using unused cycles on workstations and frequently arrive on Exabyte or other low cost media.

Data export is primarily at the DST level, although there is a growing trend which favours the export of Ntuples. Data export in particular is a non-trivial operation, requiring the allocation of media, shipping requests and the updating of file and media databases at sending and receiving sites. To what extent networks will alleviate this problem in the LHC era remains to be seen.

2.6 FATMEN

Shortly before the startup of LEP, a working group was established to investigate the requirements of the various experiments for a file and tape management system. This working group was given the somewhat unfortunate name of FATMEN, for File and Tape Management: Experimental Needs. The mandate emphasized the following points:

[1)] This is already done for the new column wise Ntuples. Active columns are unpacked and stored in memory. Inactive columns remain packed on disk.

- The working group should also look at the software which the experiments use to locate events, the interfaces between the packages used to generate production jobs and the packages used to manage dismountable media.
- Encourage the use of commercially available software or common developments between the experiments. If a common approach proves impossible then agreed specifications for inter-package interfaces must be generated.
- Look at the problem of interfacing to tape management software that may be installed at different LEP production centers.

Some of the commercial packages that were investigated will be covered later. Unfortunately, none of these packages were suitable at the time of LEP startup.

2.6.1 The FATMEN review

The FATMEN committee investigated as many commercial packages as possible. Unfortunately, these proved without exception to be unacceptable. Most targetted only one operating system and those that offered multiple platform support did so in the most rudimentary of manners. One vendor even suggested that tape access should be offered on Monday, Wednesday and Friday on the VAX and on all other days on the IBM.

At the time of the review, the Rutherford Appleton laboratory in the UK were in the course of writing a new tape management system. This system was also to be installed at IN2P3 in Lyon, and seemed to offer most of the features that were needed for our purposes. The one feature that was not initially available was the ability to allocate tape volumes at run-time, now available via the `GETPOOL` command.

At that time, CERN was in the process of installing a tape robot. If one is to use a robot safely, some sort of tape management is mandatory. Tape volumes can be write locked by hardware. In the case of round tapes, this is via a write ring. For cartridges or cassettes, it is via a thumb wheel or tab similar to that on audio or video cassettes. Although most robots are up to the task of inserting a cartridge into a drive, they are uniformly incapable of write locking or enabling a volume. Hence, some sort of software control is required.

2.6.2 Recommendations of the FATMEN committee

The FATMEN committee made the following recommendations: [3]

- A file database system should be designed and written at CERN. Users would interact with a Zebra RZ database, whereas ORACLE would be used to maintain the master database.
- A Tape Management System should be imported from the Rutherford Appleton Laboratory in the UK.
- Servers should be used to supply data from the central systems to workstations.
- The evolution of commercially available distributed mass storage systems should be carefully followed. In particular, the IEEE Computer Society Reference Model for Mass Storage Systems and systems based on it should be studied.
- ...

The file database system is what is now known as the FATMEN package.

2.6.3 The FATMEN model

The FATMEN package was designed around a layer model. Although this is not the same as the layer model used to describe the IEEE reference model, it resembles it closely.

The model is composed of the following layers:

1. *Event Tag Database*
2. *Production Database*
3. File Database Layer
4. Network Storage Management

5. Tape Management System
6. Stage/Setup Layer
7. OS Tape Interface
8. Robot/Operator Layer

The first two layers were considered to be experiment specific, although a general purpose database which may be used to record details of production, calibration constants, detector geometry and so on is now available through the CERN Program Library under the name HEPDB. At the time of the review, none of the experiments expressed interest in what was then termed the Event Tag Database. It is interesting to note that two of the LEP experiments now use precisely such a system - the Direct Access to Data (DAD) system of OPAL and the event directories of Aleph. [2]

This, and the production database, were considered to be *experiment specific.* HEPDB [4], based heavily on the OPAL and L3 packages, is now available through the CERN Program Library to address these needs.

One of the purposes of this model was to enable different solutions at any level. This was to be achieved by defining standard interfaces, although that it was clear that it would usually involve a certain amount of *glue*

Although we have not yet described the IEEE model, it is perhaps useful to introduce some of the terms and compare the IEEE definitions with those used in the FATMEN context.

The IEEE model includes two layers which map very closely to the above model. These are the Physical Volume Repository, or PVR, which is almost exactly the equivalent of the Robot/Operator layer. The next layer is the Physical Volume Library (PVL), which encompasses the Tape Management System and the Stage/Setup layer in our model. The Stage/Setup layer consists of software that is layered on top of the host operating system which, amongst other things, provides an interface between the native tape mounting commands the Tape Management System. In the IEEE model, cooperating operating systems are supposed to conform to the PVL interface. In the opinion of the author, it is more reasonable to add a small layer, as is done with the NFS server for example, rather than expect the operating system itself to interface directly to the PVL.

The Network Storage Management layer was the subject of much discussion. In fact, we realised that the network would be omni-present, but were somewhat at a loss as to how best represent this. There is a growing trend for all of the above services to be available transparently anywhere in the network. This is not yet completely true for tape staging, but a common staging system is currently being developed at CERN which should be in production later this year.

An important feature of the above model was that the actual implementation of any one layer was fairly flexible. Thus, when moving from one platform or site to another, one staging system could be unplugged and another inserted. As the IEEE model evolves and systems that conform to it begin to appear, one could imagine replacing some of the layers with commercial software.

2.6.4 Features of the FATMEN package

The FATMEN system consists of a number of components. These include:

1. A file catalogue
2. A set of distributed servers that maintain copies of the catalogue at remote sites
3. Fortran callable and command line interfaces
4. Tools for data management and export

The file catalogue provides allows experiments to refer to their data via a device, operating system and location independent manner. The recommendation is that the naming scheme should describe those attributes of the data that are most useful to the physicist. This includes such information as

– Is the data 'real' or simulated? (Or test beam, cosmics etc.)

- What stage of the production chain does it represent? (Rawdata, DST, ntuple etc.)
- Physical characteristics, such as the beam energy, type, target, magnetic field, polarisation etc.

An example of a naming scheme is shown below:

```
FM> ls -w

 Directory ://CERN/NA44/RAWD/1991/PROT/450GEV/PB/HOR-3

 RUN0391 RUN0392 RUN0393 RUN0585 RUN0631 RUN0391 RUN0392 RUN0393 RUN0585
 RUN0631
 Total of     10 files in      1 directory
FM>
```

It is important to stress that the generic name provides many forms of transparency. The same name may be used to access data on CERNVM, Shift, VXCRNA, UVVM etc. and regardless of whether the data is on disk, 3480, 8200, DAT, optical disk, in a Shift disk pool, accessed through NFS, AFS, DFS etc. to name but a few possibilities. Today, we begin to see *location transparency* through AFS or DFS, but we still do not see the other forms of transparency, which are an important feature of FATMEN.

2.6.5 The FATMEN naming scheme

There are a few limitations on FATMEN generic names, which mainly reflect the fact that the catalogue is based on the Zebra RZ package. These are as follows:

- The length of path elements may not exceed 16 characters
- The length of the filename may not exceed 20 characters
- The total length of the generic name may not exceed 255 characters
- The names are case insensitive

2.6.6 Generic names and physical files

The last example shows *two* entries for each generic name. A single generic name may point to an arbitrary number of physical files, or to another generic name. Multiple entries under the same generic name are permitted for a number of reasons. Firstly, one may have made copies that are to be exported to remote sites. Secondly, the files may reside on different types of media. Finally, copies may exist in different data representations or file formats. For example, one generic name might point to two copies: the first on a 3480 cartridge in a tape robot, on which the data is recorded in IBM native format [2], whereas the other may be on an optical disk in so-called *exchange* format. [3]

Under this scheme it is important to note that it is the user's responsibility to ensure that a new name is used if the data is modified.

2.6.7 Command interface

The FATMEN command interface is based on Unix. Thus there are commands such as `cp, mv, ls, pwd, rm, mkdir, rmdir` and so on. In addition, there are commands to manipulate catalogue entries, such as `add` and `modify`. Finally, there are utilities, such as `copy`, which provide a high level interface to file copying, described further in the section on data export, and commands to access the data itself.

2.6.8 Callable interface

The callable interface provides all of the functionality of the command line interface together with many additional features. For example, one may sort a list of generic names according to tape number and file sequence number within the individual tapes, and so on. The callable interface is recommended for efficiency, whereas the command line interface is probably more useful for casual use.

[2] EBCDIC representation for formatted data, IBM floating point and VBS file format

[3] Fixed length records, no control words; big-endian, IEEE floating point format, ASCII code for formatted data.

2.6.9 Interacting with the FATMEN catalogue

As mentioned above, the last example shows two entries for each file. In this particular case, one copy is on conventional round tape, e.g. 3420, and the other on 3480 cartridge. When attempting to access one of these files, a decision has to be made as to which copy to choose. This decision is based on a set of rules, which can of course be tailored or overridden.

The decision is not always trivial. In the simplest case it might be a choice of a copy on disk or a copy on tape, or a copy on a cartridge in a robot versus one that has to be mounted manually. However, the situation immediately becomes more complex if the disk based copy is on a different node, and is accessed via a server. Is it more efficient to access a tape copy rather than attempt a network access? Alternatively, one might find a disk copy in so-called Zebra exchange format but a tape copy in native format (e.g. VAX floating point, little endian). Is it more efficient to use the disk copy or wait for the tape mount?

Another feature that can be explained using the above example is that of catalogue subsets. For efficiency, we could limit all searches to a specific media type or set of media types. If, for example, we have no round tape support then we might as well mask out all such entries. The same can be done for data representation and location code.

```
FM> set/media 2
FM> ls -w

 Directory ://CERN/NA44/RAWD/1991/PROT/450GEV/PB/HOR-3

 RUN0391 RUN0392 RUN0393 RUN0585 RUN0631
 Total of      5 files in      1 directory
FM>
```

The latter is particularly important when many copies of a file exist. To explain this point further, we need to understand the default selection procedure in more detail.

Unless a specific copy is requested, or the default selection overruled, FATMEN will attempt to select the best copy according to the following rules:

1. For each type of medium, FATMEN loops over all copies in turn.
2. For disk datasets, a check is made to see if the file is accessible directly (e.g. via a Fortran INQUIRE), or via a server.
3. A local copy is always taken in preference over a served copy
4. For tape datasets, a check is made to see if the volume is available (e.g. not archived) and whether the device type required is available (or served) on the node in question.

If 30 copies of a DST have been made and sent to outside institutes, one can avoid making redundant queries to the Tape Management System by assigning location codes and masking off a subset of the catalogue appropriately.

2.6.10 Creating new data with FATMEN

It is certainly more complicated to create new data that is catalogued in FATMEN than to access existing data. However, the following model, which is designed for production chain jobs, is relatively straightforward to implement.

It is assumed that the generic name of an output file differs only in one directory name from that of the input file. Thus the input file

```
//CERN/NA44/RAWD/1991/PROT/450GEV/PB/HOR-3/RUN0391
```

might lead to the output file

```
//CERN/NA44/DST/1991/PROT/450GEV/PB/HOR-3/RUN0391
```

Many of the fields in the output entry are generated automatically, such as the node name on which the job runs, the date and time, the user name and so on. Other fields, such as the Zebra format, record length and so may be copied from input or taken from a model entry, cf the so-called model DCBs on the DESY IBM system.

Assuming that the file is to be written to tape, the following operations may be required:

1. Allocate a new tape or a new file on a multi-file volume
2. Issue an output staging request
3. Write the data
4. If the job is successfully then
   (a) Request that the data be copied from the staging pool to tape
   (b) Write lock the tape
   (c) Add a comment (tag) to the TMS for this volume
5. If the job is unsuccessful then the volume (or space) is freed and the catalogue entry dropped

To be more explicit, one might wish to allocate a tape volume from a certain pool, e.g. `XX_FREE`. After writing the output dataset this volume might be moved to another pool, e.g. `XX_DSTS`.

2.6.11 Data export using FATMEN

Data export can be performed through FATMEN in a number of ways. Firstly, one may use the routine `FMCOPY`, or the corresponding command `COPY`. This provides numerous copying options, including the use of `STAGE CHANGE`, full Zebra I/O (thus permitting data representation and file format conversion), network copying (both TCP/IP and DECnet) and finally transmission via satellite, as described below. Alternatively, one may use the FATMEN primitives directly, as has been done by OPAL for the tool `EXPOCART`.

2.6.12 Data export using CHEOPS as transport

CHEOPS [5] is a project that uses the Olympus satellite to transmit physics data to remote sites. Transfer requests are made through FATMEN, either by using the command `COPY` or through the `FMCOPY` library routine.

The transmission is asynchronous. During the day, requests for copies are received and the data pre-staged to disk. The data is then transmitted at night.

When the user makes a copy request, the input data are verified by FATMEN. This includes ensuring that all parameters required by CHEOPS are supplied, either explicitly or implicitly via an appropriate catalogue entry.

An entry is added immediately to the FATMEN catalogue with the comment field set to something like

```
Copy request queued to CHEOPS on 930725 at 1315
```

At the same time, a copy request is queued to the CHEOPS server. The CHEOPS server processes the request and sends back a reply which is processed by the FATMEN server. This reply might indicate success, in which case a request ID is returned, or an error condition. In both cases the comment field of the corresponding FATMEN entry is updated, so that the user can track the progress of his request.

Finally, once the transmission has completed successfully, a further report file is sent from the CHEOPS server to the FATMEN server. At this time, the original comment supplied by the user is restored in the FATMEN catalogue.

About 37 GB were transferred by CHEOPS in July 1993. Nearly all of these data was sent to Helsinki, on behalf of the DELPHI collaboration.

The CHEOPS protocols do not depend on a satellite and could be generalised to use any medium, such as a normal terrestrial network. Similarly, the code in FATMEN to handle asynchronous copying is independent of CHEOPS and could equally well be used with cooperating tape copy stations etc.

2.7 Requirements for LHC and SSC

The data handling requirements for experiments at the LHC or SSC are expected to be significantly larger than those of the current generation. For example, some 10-

100MB/second are expected at the 3rd level trigger at the LHC. Although these rates sound large, they can, in theory, already be handled today. The E791 experiment at Fermilab built a data acquisition system that wrote to 42 Exabyte 8200 drives in parallel [6]. This system achieved an average throughput of 228KB/second, which is certainly very respectable when compared to the theoretical capabilities of the Exabyte 8200 drives, which are 246KB/second. Using this system, the experiment managed to acquire 50TB of data on 24K 8mm tapes, during their run in 1991.

Using similar ideas, one could cope with the anticipated data rates from LHC or SSC by using technology such as the Ampex D2 drive, which has a theoretical throughput of 14MB/second, in parallel.

We expect several (tens of) PB (10^{15} bytes) of data per year. It is already possible to build a multi-PB library, using technology such as assembled by E-Systems [7].

In summary, although the hardware requirements for the LHC would appear to be very demanding, we could, given sufficient money, build working solutions today. However, the software requirements are much less clear.

2.8 Data management in the LHC era

One of the most critical problems for the LHC era (in the area of data management) is the question of data access. Should access be at the file, event or byte level, where we expect some 10^6, 10^9 and 10^{15} objects respectively. Although some people argue that this is a database problem, it is clearly a non-trivial one. Current database technology is not capable of handling such large numbers of objects, particularly when good performance is required. The D0 experiment at FNAL already have of the order of 10^6 files. There are no intrinsic reasons why today's methods will not work with an order of magnitude more files. However, an increasing number of people argue that access by filename is inappropriate and that access via high level attributes is what is really required. This remains a largely unsolved problem.

However, it is clear that we do not wish to continue to write and maintain our own solutions for the next generation of experiments, but would prefer to obtain standard software, whether *de-facto* or *de-juro*. Open questions include distributed data, processing power and people versus distributed people, centralised processing and data [8]. Given the enormous volumes and the rapid decrease in cost of processing power, the latter solution would appear to be the more logical. Carried to its ultimate extreme, it implies keeping the data as close as possible to the experiment itself, and installing large processor farms there. This would be largely a continuation of an existing trend.

3 Review of existing storage systems

The developers of the IEEE MSS Reference Model recognised a clear need for prototype systems that were deployed in real environments. All of the following mass storage systems have been commercialised and have greatly influenced the development of the model.

A common feature of these systems is that they are designed to support supercomputers. In addition, the applications are very different to HEP. Although the data volumes are large, the data rates are relatively low.

3.1 The origins of the IEEE Mass Storage Reference Model

The Mass Storage Reference Model finds its origins at a workshop held at NCAR in 1983. Although there had been several Mass Storage Symposia prior to this workshop, the idea of a generic model was first formulated at this meeting. The objectives of the workshop were to assemble a group of mass storage specialists whose task was to document ideas and experience from existing systems and to present these at the 6th IEEE Mass Storage Symposium. It is interesting to note some of the topics of discussion, which included *centralised* versus *distributed* systems, the integration of the system with the host operating system and whether the system or user should be responsible for the placement of files in the storage hierarchy.

3.2 The Los Alamos Common File System

The Los Alamos Common File System (CFS) [9] is probably the best known mass storage system. It is marketed under the name DataTree, by General Atomics. CFS was developed in 1979 and is still used in production at many sites today.

CFS uses an IBM (or compatible) mainframe, running MVS, as storage server. Client systems are available for a large number of platforms, including COS, Unicos, CTSS, NOS, VMS, Unix, VM and MVS systems. CFS was originally developed in 1979 and has since been installed at some 20 sites, including the European Centre for Medium Range Weather Forecasting in Reading, UK. It is mainly written in PL/1.

CFS consists of a mass storage processor connected to a high speed network. The processor maintains a directory and journal file online and migrates files between a disk cache and offline storage. Migration depends on file size and last access. Files that are above a certain threshold in size are never stored in the disk cache but always offline.

To avoid disk fragmentation, disks are separated into classes with varying allocation units. This small files will be placed on disks with small allocation units and so on.

This is similar to the cluster size attribute for disks on VAX/VMS systems. The cluster size is the number of 512 byte disk blocks that are allocated at a time. The default is 3, which is clearly unsuitable for large files, such as those on the staging disks. On the other hand, the cluster size of 250 blocks which is used on the staging disks would be completely inappropriate for `home directory` style files, as is shown by the following example.

```
AXCRNB? create disk$stage:[cnsupport]jamie.test
This is a test to show how wasteful a large cluster
size can be for small files.
 Exit
AXCRNB? dir/siz=all disk$stage:[cnsupport]jamie.test

Directory DISK$STAGE:[CNSUPPORT]

JAMIE.TEST;1             1/250

Total of 1 file, 1/250 blocks.
AXCRNB?
```

Files stored in CFS are accessed via a Unix-style pathname. Files may be manipulated via special CFS client software, which provides commands such as GET, MOVE, REMOVE etc. In addition, an ftp interface has been developed, which is particularly interesting in today's environment, as it removes the problem of client software installation and support on multiple platforms.

One of the key features of CFS is that of device transparency. It provides access to data in a device independent manner, but also permits old technology to be replaced transparently. This means that the system can automatically migrate data off obsolete media onto newer media in the background.

It is almost guaranteed that the predominant medium at the startup of LHC will be obsolete during the lifetime of the machine. Given the volumes of data involved, the possibility of retiring old media is almost a requirement.

3.2.1 A user's view of CFS

A user typically sees both a local file system and the CFS. Files may be copied between the local file system and CFS. Once in CFS, the files may then be retrieved by any other node running a CFS client. Users may also specify attributes such as access frequency, which influences the decision on eligibility for migration.

3.3 The NCAR Mass Storage System

Like CFS, the NCAR MSS [10] is also based upon a central dataserver running MVS. It was developed at the National Center for Atmospheric Research in Boulder, Colorado after experience with CFS. Again like CFS, it is mainly written in PL/1.

The NCAR systems includes a feature which is frequently emphasised in the IEEE model, namely the separation of control and data. Control messages are passed over a standard network, whereas the data transfer is made over a so-called fast path, optimized for bulk transfer. In today's environment one might use standard Ethernet for control, but FDDI or HIPPI for bulk data transfer.

The NCAR system provides automatic repacking of tape volumes. As files are deleted, either explicitly or via an expiry mechanism, holes occur on tape volumes. Eventually the data is rewritten to a new volume and the original volume freed.

The system also protects against media aging, by reading tape volumes at random. Should errors occur, the data are copied to a new volume without user intervention.

3.4 The NASA-Ames Mass Storage System

The NASA-Ames MSS-II [11] system is centered on Amdahl storage server running the UTS operating system. Its goal is to appear to the end user as just another Unix system with an infinite supply of very fast disk space. This system, the first we have described that was written for Unix, incorporates the following components:

- A striping file system (equivalent to Berkeley RAID-5)
- A hierarchical storage manager for Unix
- A volume manager, e.g. a Tape Management System
- A striped network

It is also important in that it is hidden **under** Unix. That is, the user does not have to learn a new set of commands, but can use standard Unix commands. In addition, the system is completely transparent to high level languages, such as Fortran or C.

The migration component will migrate files under two conditions. The first is periodic migration, which will typically be invoked using the **cron** utility. The second occurs when *file system full* condition occurs. Migration only affects data blocks - the inodes and directory structure remain intact. Demand restore occurs when the users issues an **open** for a file whose data blocks are not on disk. I/O will only block if a block that is not already on disk is referenced. As data is restored, it is delivered directly to the application from the system buffers in parallel to the restoration to disk.

Tapes are written in ANSI labelled format and contain only the files of a single user. This provides a convenient means of dealing with files which have not been accessed for an extremely long period - the associated directory structure is deleted and the appropriate tapes are mailed to the user in question.

3.5 The Lawrence Livermore Storage System

The Livermore Storage System is more commonly known by the name Unitree. Unitree is in fact a port of the Livermore system from their NLTSS operating system to Unix. Unitree is also marketed by General Atomics (since taken over by Open Vision). NLTSS is the Network Livermore Time Sharing System, a follow on the LTSS. In NLTSS, there is a single mass storage server, running under UTS (Amdahl's version of Unix). The system is coded in C.

Given the relatively wide-spread availability of Unitree, the remaining discussion is based upon the commercial product.

Unitree is now available through numerous suppliers, including CDC, CONVEX, DEC and IBM. Unfortunately, the current versions that are available in the market place do not fulfill many of the early promises of the company. Most importantly, there are significant performance problems in a number of areas, particularly NFS access, and there are a number of design limitations which prevent its use at a large site such as CERN.

Firstly, the number of tape volumes is limited to 10K, although some vendors have increased this to 100K. The CERN Tape Management System currently tracks well over half a million volumes. The meta and tape catalogues cannot be backed up online, requiring up to 3 hours per day when the system is unavailable. In addition, the performance of commands such as `ls` is unacceptable. One site quotes roughly one hour to perform an `ls -l` of one thousand files.

3.6 System Managed Storage

System Managed Storage is a response to a **SHARE**[4] white paper of 1979. SMS was not announced until 1988, as the `Data Facility Storage Management Subsystem`. It was initially available under the MVS operating system only, but some components have since been made available under VM/CMS. Many of the problems cited in the white paper were clearly due to some of the less user-friendly features of IBM operating systems. Nevertheless, some extremely important features, such as *device transparency*, *storage classes* and *transparent device conversion* where high on the list of requested features.

3.7 The FATMEN review

All of the systems described above were reviewed by the FATMEN committee. Unfortunately, none of them appeared to satisfy our requirements, for the reasons cited below.

- CFS requires an MVS based server. The CERN MVS system was scheduled for termination at the end of June 1989. More importantly, CFS required a proprietary network such as Ultranet or Hyperchannel, although ftp access was possible via a gateway server. In addition, there was no CFS client for VM/CMS systems.
- The NCAR system again required proprietory network protocols. In addition, it had not been installed at a single site outside of NCAR, although it has since been acquired by one of the NASA sites.
- NAStore unfortunately runs only on Unix. We also needed to provided client support for VM/CMS and VAX/VMS systems.
- Unitree was not yet available. Again, it only targetted the Unix world.
- IBM's SMS looked interesting, but was IBM only.

In addition, all of the above products, with the obvious exception of SMS, only supported the STK silo. CERN had just started a joint project with IBM on their cartridge robot.

3.8 Recent developments

Many of the sites responsible for the development of the above products have continued to develop Mass Storage Systems. In particular, it is worth mentioning the recent developments at Los Alamos and the National Storage Laboratory hosted by Lawrence Livermore National Laboratory.

3.8.1 Los Alamos High Performance Data System (HPDS)

HPDS is a fourth generation storage system designed to meet the storage and access requirements of Grand Challenge problems. These applications typically run on supercomputers and massively parallel machines and generate gigabytes to terabytes of output. HPDS is designed to store up to petabytes of data, deliver data at tens of megabytes per second and handle files up to terabytes in size. Under HPDS, storage devices are directly attached to the network and data transfer passes directly from the storage device to the client, without passing through an intermediary server. Client access is based on an ftp-like interface called *Data Transfer Interface* (DTI). DTI differs from ftp in that it permits

[4] The IBM users' group

partial file transfers and appends. Future enhancements include powerful scientific data management tools and meta data handling.

HPDS has a limited lifetime and will be superceded by HPSS, to be developed at the National Storage Laboratory.

3.8.2 The National Storage Laboratory

The National Storage Laboratory is located at Lawrence Livermore Laboratory in California. To some degree, it can be considered a proof of concept laboratory where new storage ideas are put into practice. There are two main areas of development, namely NSL Unitree and HPSS.

NSL Unitree is based on Unitree 1.7 with enhancements, which include third party data transfer, as in HPDS, and multiple dynamic hierarchies. The latter is a concept similar to storage classes introduced by SMS and is described further below.

3.9 Dynamic Hierarchies

The current storage paradigm handles a fixed hierarchy, which often consists of disk and tape, or perhaps disk, robotic tape and manual tape. However, this hierarchy does not exploit the features of different devices, such as solid state disk, RAID, 3480, 8mm, DAT, DD-2 etc. A more general approach is to build a multi-level hierarchy and define different migration paths for various types of data. For example, large, infrequently accessed data might migrate directly to DD-2, whereas more frequently referenced data might migrate to optical disk. In addition, this scheme should be flexible enough to permit the removal and addition of old or new media types. To remove a media type, e.g. 3420 tape, one would edit the appropriate configuration file and perhaps add a new type at the same time. The system would then ensure that the data were transparently migrated from the old media, which could then be removed from service.

3.10 Network File systems

3.10.1 NFS

The Network File System or NFS, originally developed by Sun, is almost certainly the best known distributed file system. NFS is based upon a stateless protocol, using remote procedure calls over UDP.

NFS *servers* must explicitly `export` those file systems that they wish to be network accessible. *Clients* must mount these file systems, or use the automounter, which automatically mounts file systems as needed and then dismounts them after a certain period of inactivity. The mount point may, and often does, differ from client to client, leading to considerable confusion.

NFS security is based upon Unix user and group IDs, which mean that it is only secure within a well-defined and controlled network.

NFS clients and servers exist for many systems, including Unix, VMS, VM, MVS and Novell (in some cases, such as VM/CMS, not all NFS functionality is supported. VM/CMS, for example, does not provide an NFS client).

3.10.2 The Andrew File System

The Andrew File System, or AFS, was developed at Carnegie-Mellon University, Pittsburgh. It is marketed by Transarc Corporation and has been selected by the Open Software Foundation (OSF) as the basis for its Distributed File System (DFS).

One of the important design considerations for AFS was that it be compatible with NFS. This is performed by the AFS-NFS exporter, which permits AFS file systems to be accessed by NFS clients through a gateway machine. For example, on VXCRNA the logical name `AFSCERN` points to a NFS mount of `/afs/cern.ch/user`.

From a user point of view, one of the most important features of AFS is the `global name space`. Thus, the file `/afs/cern.ch/user/j/jamie/fatmen/fatmen.car` can be accessed transparently from the SSCL in Texas, CERN, or indeed any other node that can see the `cern.ch` cell.

Before describing what is meant by the term `cell`, it is worth mentioning an additional feature that is much appreciated in the wide area. This is the concept of local caching.

As can be seen by the following example, the first access from Texas to a file at CERN involves a small time delay. Subsequent accesses, however, are almost indistinguishable in terms of speed from those performed locally. Note that there is an improvement in access time even for local access, for the same reason.

```
zfatal:/afs/cern.ch/user/jamie/fatmen (543) time head -1 fatmen.car
+TITLE.

real    0m0.39s
user    0m0.01s
sys     0m0.03s

zfatal:/afs/cern.ch/user/jamie/fatmen (545) time head -1 fatmen.car
+TITLE.

real    0m0.06s
user    0m0.02s
sys     0m0.02s

synergy.ssc.gov$ time head -1 /afs/cern.ch/user/j/jamie/fatmen/fatmen.car
+TITLE.

real    0m7.36s
user    0m0.00s
sys     0m0.12s

synergy.ssc.gov$ time head -1 /afs/cern.ch/user/j/jamie/fatmen/fatmen.car
+TITLE.

real    0m0.05s
user    0m0.02s
sys     0m0.02s
```

AFS cells An AFS *cell* is a group of servers and clients in a single administrative domain. AFS cell names are typically those of the ftp domain, e.g. `cern.ch`.

Operating system dependant pathnames One obvious problem with a global naming scheme is the handling of operating system dependant code. The files that one would normally put in the `bin` subdirectory of one's home directory are likely to be operating system specific, unless they are shell scripts.

This is handled in AFS by the use of `@sys` in the pathname. This is automatically translated by AFS into an operating system specific path. On my RS6000, this translates to `rs_aix32`. On an HP machine running HP/UX version 9, this would translate as `hp700_ux90`. Thus, by linking the `bin` directory to `@sys/bin`, the correct binaries will be automatically selected.

3.11 AFS and physics data

The global naming scheme of AFS would appear to be attractive for accessing physics data. Although AFS cannot itself improve the network bandwidth, local caching will certainly result in lower network load and better response time for repeated accesses to the same files than NFS. As such, it is well suited to accessing small amounts of test data in read only mode. It is not well suited to handling files which are frequently updated, as this would result in equally frequent cache updates. It also does not offer any improvement over NFS for bulk data access. Both of these problems can be solved by using a library routine which bypasses the cache.

3.12 Integrating AFS with hierarchical storage systems

One feature that is lacking from standard AFS is hierarchical storage management. Two sites have done interesting work in this area, namely the Pittsburgh Supercomputer Center and the University of Michigan. The approaches taken are quite different and are described briefly below.

3.12.1 The Institutional File System (IFS)

The Institutional File System was developed at the University of Michigan [12]. It is based on AFS, over which it offers a number of enhancements. These include new server platforms, intermediate servers, new classes of clients and extensions to the authentication mechanism.

The main new server platform is the IBM 390 architecture, running under MVS, VM/CMS or AIX/370. This permits IFS to exploit the traditional strengths of a mainframe, namely high I/O capability, large disk capacity and main memory, high reliability and general mainframe utilities. The importance of many of these items was clearly greater when IFS was designed in the later 1980s.

A more important extension is the addition of hierarchical storage management, provided on the mainframe by IBM's DFHSM. In addition, IFS have made a number of enhancements in the area of caching. The most significant of these is the concept of *intermediate caching servers.* IFS supports tens of thousands of workstations and PCs and it was felt unrealistic to serve such a high number systems using a single or small number of traditional AFS servers.

Further enhancements in the caching policy are planned, based on the following observations.

- After a file block has been accessed, the chances that it will shortly be reaccessed are high. Caching the block exploits temporal locality
- The chances that the subsequent block will be accessed is also high. Prefetching exploits this probability
- When a file on an archived volume is retrieved, other files from the same volume are frequently retrieved shortly after. Restoring all files from a tape exploits this. `This does not necessarily hold for HEP. This feature was implemented in the VAX staging package used by FATMEN but did not help as anticipated at DO, FNAL`
- Some blocks are more important than others. One should weight blocks appropriately in the flushing algorithm. e.g. directory blocks should be retained over data blocks
- Blocks should also be weighted in relation to their *acquisition cost.* e.g. blocks that were expensive to obtain, e.g. read from tape, should have higher priority than those that were cheap to obtain
- Similarly, some blocks are more expensive to flush, as they must be written to a slower storage device
- Once a user lists the contents of a directory, he frequently attempts to access a file in that directory shortly afterwards. Such `hints` should also be taken into account
- Some blocks within a file are more popular, e.g. `head` and `tail` blocks

3.12.2 Pittsburgh Supercomputing Center

Prior to migrating to AFS, the Pittsburgh Supercomputing Center used CFS to provide mass storage services. AFS has been extensively enhanced to meet their requirements for a mass storage system as described below [13].

Firstly, AFS has been enhanced to provide hierarchical storage management. For this purpose, AFS was modified to support *multiple residencies.* This permits a copy of a file to exist in up to 32 places. This has the side effect of enhanced reliability. If one component of storage system is down, a request for a given file will be satisfied from highest priority storage system that is available. Data migration is performed between storage systems according to attributes in a database. These attributes include characteristics such as desired file size distributions amongst the different levels of the hierarchy. Media types can be replaced transparently by issuing only two commands.

For practical reasons, multiple AFS servers are required. Currently, each server can only support 52 GB of disk space. To avoid having to connect a robotic device of each type to every AFS server, the servers have been modified to communicate between themselves via RPC to provide shared storage services. Supported servers currently include Maximum Strategy RAID arrays, HP optical jukeboxes, Crays running Data Migration Facility and RS6000s running Unitree.

3.13 OSF/DFS

The OSF Distributed File System is based on AFS version 4.0. There are a number of differences between AFS and DFS, which include the following. DFS is currently only available for IBM RS6000 machines (both client and server). The RS6000 server may use either the standard AIX journaling file system or Episode. Episode is the standard file system for DFS and provides the basic functions that are required, e.g. ACLs. In addition, it is also log-based, which is important if one is to avoid length restarts after server crashes. DFS path names start /... rather than /afs. DFS and AFS can coexist, but are two separate file systems with two separate caches.

Is the global naming scheme of AFS or DFS with its caching capabilities and hierarchical enhancements such as those made at the University of Michigan the answer to our problems? Unfortunately, although extremely attractive for *home directory* style files, such a solution is probably not optimal for physics data files, for reasons that are discussed below.

3.14 Home directory file services

Home directory file services include *backup/restore*, *migrate/recall* and *archive/retrieve*. Surprisingly, these functions are often confused.

3.14.1 Backup and restore

Backup and restore can be further subdivided into *disk* (or filesystem) and *file* backup/restore. Disk backup is performed to permit disaster recovery. To a large degree, it can be avoided by using techniques such as RAID. Furthermore, it can be simplified by using *dataless* workstations. Firstly, one should ensure that the workstations are running from standard system disks. Any local tailoring that is required should be done using a script from a standard starting point. This permits one to backup only the *master* copy of the system disk and not the copy on each (remote) workstation.

File backup is performed to permit users to recover accidentally deleted or corrupted files. As such file restore must clearly be user driven. An analysis of restore requests on the central VAXcluster VXCERN at CERN shows that the provision of version numbers in the VMS file system virtually eliminates the need for file restore. Unfortunately, Unix does not provide version numbers. However, one may achieve a similar effect by using code managers which keep copies of multiple versions of a program. In addition, techniques such as the *trash-can*, where deleted files are stored for a few hours, permit the recovery of files deleted by mistake. Finally, one should ensure that easily recreatable files, such as *a.out, core, .o* etc. are excluded from the file backup.

3.14.2 Migration and recall

File migration provides the user with virtually unlimited disk space. Unused files move from faster, more expensive media to slower, lower cost media. Recall must clearly be transparent to applications and user programs, but both migration and recall should be user-driveable. This permits users to explicitly migrate files that they know will not be accessed for a long time, e.g. if they are about to leave on sabbatical and to recall files that will soon be accessed, e.g. when they return. If file migration is not provided, people often misuse file backup and archive facilities. It is clear that many of the concepts of SMS are required, whereby files are migrated to an appropriate level in the storage hierarchy, as determined by their storage class.

3.14.3 Archiving and retrieval

Archiving has a number of features that differentiate it from migration. The most significant is that archived files belong to a different name space. Some people argue that this is an advantage, although this can of course be achieved using a simple script built around the migrate command. This script would demand migrate the file and then *mv* it to a different filesystem or directory or rename it to a dot file, i.e. one that is not visible to a simple `ls` command. Thus, provided a good migration system is available, the need for archiving is considerably reduced. Some sites, such as the Rutherford Appleton Laboratory in the UK, have maintained a policy of removing archival systems in favour of migration systems.

3.15 Open Storage Manager

The Open Storage Manager is a set of three products from Lachman technology. The family of products are built around the IEEE MSS reference model and comprises
1. Transmigrator, a file migrator
2. Conservator, a storage system
3. Meditator, a distributed removable medium and library management package

3.15.1 Transmigrator

Transmigrator is a software package that provides file migration services for existing Unix file systems. Files may be migrated in any of three ways:
1. Explicitly, i.e. at the request of a user.
2. On demand, i.e. of there is a lack of disk space.
3. Periodically, e.g. from a cron job.

When a file is migrated out of the normal file system, a so-called *stub-file* remains. This stub-file contains the first data blocks of the file plus a *bitfile-ID* which points to the migrated copy. Obviously, there are thresholds on the minimum file size that a file must have before being eligible for migration. In addition, there are thresholds for the minimum age of a file. By default, these thresholds are set to 10KB and 1 day since last access.

Watermarks Transmigrator uses three watermarks. In addition to the traditional high watermark, at which point migration is initiated, and low watermark, at which point migration terminates, there is a so-called *prestage* watermark. This watermark is lower than the low watermark. After reaching the low watermark, transmigrator may optional migrate additional files down to the prestage mark. These files remain on disk in their entirety. However, should disk space be urgently required, it can now be released very rapidly simply by creating the stub-files - the data has already been migrated into the backing store.

Symbolic attributes Two symbolic attributes are associated to a migrated file. These are the storage group and migration path. These attributes permit different files to be migrated into different stores, and control the subsequent migration within those stores. This can be useful when the attributes or access patterns of certain files are known. Many of the files that are migrated will be log files, which are rarely, if ever, recalled. One may then migrate these files directly to a low level in the hierarchy onto low cost devices.

3.15.2 Conservator

The Conservator is a storage repository that is used by Transmigrator when migrating and recalling files. It is not required by Transmigrator, which can migrate to and from any mounted file system.

Conservator supports multiple internal hierarchies. That is, certain files may be migrated to one group of storage devices and others to a different, potentially overlapping group of devices.

3.15.3 Mediator

The mediator handles dismountable volumes and robotic libraries. In terms of the IEEE MSS RM, it provides the functions of both the PVL and PVR.

The mediator provides a number of timers to optimise operation. These are the *wait-timer, hog-timer* and *release-timer.*

When a volume is released, it is maintained online for *wait-timer* seconds. If a new request for the same volume arrives within this interval, it will be satisfied immediately.

To protect against a given user repeatedly remounting the same volume, the wait-timer is turned off after *hog-timer* seconds from the time when the volume was first released.

3.16 ADSTAR

As one approaches the Adstar building in San Jose, California, the meaning of the company's name is made clear - Advanced Storage *and Retrieval.* These last two words feel strangely reassuring.

ADSTAR is a component of IBM. It has offices in Tuscon, Arizona, which make tape products, and in San Jose, where disk products and storage software are produced. ADSTAR are currently represented at the PVR working group meetings.

The ADSTAR Distributed Storage Manager provides backup and archive services for local, NFS and AFS file systems. Space management (migration) is currently being added. ADSM clients are already available for MVS, VM/CMS, many Unix platforms and Novell. Additional clients are planned for other Unix platforms and VMS systems. The servers are also being ported to Unix platforms. ADSM supports many concepts from SMS, such as dynamic hierarchies and transparent media retirement.

In addition to command line and Motif interfaces, ADSM offers a library interface. This interface is potentially extremely interesting for the storage and management of HEP data.

3.17 A possible solution to physics data storage

One of the big problems of using AFS together with hierarchical storage management for physics data is that all data must pass through the file system. Not only is this inappropriate for certain files, such as rawdata files, but it also gives problems due to the sheer number of files involved. Currently, experiments have between 10^5 and 10^6 files. These numbers are already large for Unix file systems. Future experiments can expect up to 10^9 files, although the actual number is likely to be somewhat smaller. Today's file sizes are limited to roughly 180MB so that they fit conveniently on a 3480 cartridge. Tomorrow's media are likely to have capacities well in excess of 1GB, so that the actual growth in the number of files is likely to be somewhat reduced.

A feature that is frequently requested by Unitree users is the ability to bypass the disk cache for certain types of files - typically large data files such as those used in HEP. CFS read and wrote files larger than a certain threshold directly from/to tape - in 1979!

In addition to the support for multiple dynamic hierarchies and retirement of media, ADSM also permits meta-data to be associated with files. Thus, one could consider using the ADSM product to manage physics data in one of three ways:

1. By modifying the existing FATMEN system to use the ADSM API to store and retrieve data
2. By replacing the existing FATMEN catalogue with the ADSM internal catalogue and a small amount of code based on the API
3. By using a conventional Unix file system instead of the FATMEN catalogue, with meta-data stored in dot files

In all cases, ADSM would take on the functions of our current Tape Management System, Staging systems and Robot control.

4 The IEEE MSS Reference Model

4.1 History

The IEEE MSS Reference Model grew out of the initial generic model first presented at the 6th IEEE MSS symposium. The model is now approaching version 5, and it is planned to freeze the model soon and begin work on standardising the various layers of the model. Unfortunately, there are still a number of areas of disagreement. However, it is likely that the model will mature bottom up, the first standard being for the PVR.

It is frequently pointed out that there are two sorts of standards: *de facto*, like Sun's NFS, which is widely available and *de juro*, like OSI, which is essentially unused. We would clearly like a standard at least as interoperable and available as NFS.

4.2 The Storage System Standards Working Group

In 1989, the IEEE Computer Society Technical Committee on Mass Storage Systems and Technology created a working group, known as the Storage System Standards Working Group or SSSWG. The goal of the working group is to convert the model into a set of standards for submission to the IEEE and ANSI.

The immediate goal was the decomposition of the model into interoperable functional components that could be offered by vendors as commercial products. The long term goal was to define interoperable standards which define the software interfaces and protocols associated with each of the architectural elements of the model.

The members of the SSSWG come from vendors, such as DEC, HP, IBM, StorageTEK, Convex plus user sites, such as Lawrence Livermore National Laboratory, the University of Michigan and CERN. To acquire voting rights, at least 3 out of the previous 6 meetings must be attended. Up to now, all meetings have been held in the US but it is planned to start a series of workshops in Europe, the first of which will be held at CERN later this year.

4.3 Important architectural features

The following features figure prominently in the design of the model.

1. Location independence
2. User and system oriented file identifiers
3. Separation of control and data paths
4. Applicable to all sizes of storage systems

4.4 Overview of the components of the model

The model is built out of a number of functional components. These are as follows:

- **Movers** are responsible for copying data to and from storage media and requesting the transmission of data from source to sink.
- A **Physical Volume Repository** is responsible for storage removable media containers, such as tape cartridges or optical disk platters, and mounting these containers onto devices through robotic or human agents.
- A **Physical Volume Library** maps storage media, or physical volumes, to movers. The PVL provides a client interface that permits physical volumes to be mounted. The PVL maps these volumes to cartridges.

– A **Storage Server** provides mechanisms to compose stores from discretely addressable storage spaces and translate access requests to these stores into a set of requests to the appropriate mover and/or PVL if the associated volume is not mounted.

4.5 A personal view of the model

I feel that it is important to use terms whose technical meaning is as close as possible to that of the English words used. In that respect, one would define a **Physical volume** as being the smallest entity that the **Physical Volume Repository** can address. On top of such objects, one might build **logical volumes**, which would be managed by the **logical volume library**. These definitions can be explained by a simple analogy with books. A **volume** may contain several books, alternatively a book may be spread across many volumes. In this analogy, the volume, which is the smallest thing that we can pick up in a book shop, is the **physical volume** of our model. The **logical volume** is the equivalent of a book.

Logical to physical mapping can be used to explain existing objects, such as those found on VAX/VMS systems. A **volume-set** is a bound set of disks that appear to higher levels as a single entity. In this respect, the *logical volume* is composed for two or more *physical volumes.* A **shadow-set** is also a bound set of one or more disks, however its appearance is completely different to higher layers! In this case, the data is replicated on the different physical volumes, giving reliability and performance improvements (at least on read). A further possibility is that of a **stripe set** or even of a striped, shadowed volume set.

In addition to the advantages expressed above, logical volumes also have benefits in terms of transparent migration from one physical volume to another and for caching purposes. At CERN, we have used single member shadow sets on VXCERN for many years. When a hardware problem is suspected, a spare disk is added to the shadow set. Once the *catch up copy* has completed, the original disk can be removed and repaired. Thus we have performed transparent migration from one physical volume to another. The same technique could also be used to perform transparent media migration. The local volume, which today might map to a complete physical volume, may well be moved to much higher capacity media in a many that is completely transparent to the user. This feature will certainly be required for the LHC era. Other benefits include cached, striped logical volumes. High capacity media often fails to bring with it an appropriate increase in throughput. If one requires that a complete volume be processed in less than 100 seconds, to avoid problems with drive and volume contention, one finds that even striping is not sufficient to solve the problem. Logical volumes help by keeping the size of data that is to be moved at a manageable level. Finally, one could move logical volumes according to access patterns. Those that were frequently accessed together could be moved transparently so that they were on a single physical volume.

4.6 The official view

Unfortunately, although the IEEE model supports the concepts described above, the official names are *cartridge* for the smallest object that one can address and *physical volume* for our *logical volume.*

5 Future developments

A number of developments are foreseen to enhance FATMEN in the short and medium term. In the long term, it is hoped that the IEEE Mass Storage work will develop sufficiently so that standard solutions can be used.

5.1 The short term future of FATMEN

The main short term changes concern usability. Originally, it was envisaged that the main use of FATMEN would be via the Fortran callable interface from production programs. To facilitate access to data, a command *fmln* will be provided which will make access to a catalogued file, regardless of its location, as simple as creating a Unix link.

For example, before running a program that processes a data file on Fortran logical unit 1, one might make a link as follows:

```
ln -sf ~jamie/data/fxfile.dat fort.1
```

The same program could then access a file catalogued in FATMEN if the following command were first issued:

```
fmln //cern/13/prod/data/ldre/cc02zkxu fort.1
```

Although data cataloguing is normally performed only by a small number of production managers, a similar command could be provided to simplify this task, as shown below:

```
fmcat //cern/13/prod/data/ldre/cc02zkxu vid.[fseq] [options]
```

5.2 The medium term future of FATMEN

In the medium term, one could envisage Fortran 90 and/or C interfaces, which would use standard language features, such as derived data types of structs, rather than Zebra banks. More importantly, the area of data export could be improved, e.g. by the provision of an asynchronous data export service along the lines of the CHEOPS project but possibly using terrestrial networks.

In addition, a standard *event catalogue*, such as that used by Aleph, H1 or OPAL could be provided and integrated with FATMEN.

To facilitate the use of standard conforming products, as they emerge, the declared interfaces of the components of the FATMEN model could be made conformant to the IEEE Mass Storage Systems Reference Model. This would allow us to replace our existing Tape Management System with a commercial solution, for example.

5.3 The long term future of FATMEN

Unfortunately, it is unlikely that the IEEE model will ever solve all of our needs. For example, the *names server* component is currently outside the scope of the model. Other important areas that are not presently addressed include the handling of files with internal structure (the IEEE model treats a file as just a stream of bits) and the whole question of data management.

However, it is clear that the IEEE work will never address issues as HEP specific as Zebra FZ initialisation. Consequently, a (hopefully thin) layer of HEP specific code will always be required. Nevertheless, it is extremely important that the HEP community provides adequate input to the Storage System Standards Working Groups.

6 Summary

High Energy Physics has a data management problem that is at least the equal of that of any other field. Future experiments will present even more challenging problems and it is essential that we start trying to understand these problems now. As data management touches many fields, we should adopt a model on which to build. The logical choice would appear to be the IEEE reference model for Mass Storage Systems. This model already addresses many areas that we know are of critical importance, such as *dynamic hierarchies*, transparent transition from old to new technologies etc. Rather than work in a vacuum, we should work closely with the IEEE Storage System Standards Working

Group to ensure that any standards or products that appear will cope with the needs of HEP.

References

[1] S. Coleman, R. Watson, R. Coyne, and H. Hulen. The Emerging Storage Management Pardigm. In S. Coleman, editor, *Proceedings of the 12th IEEE Symposium on Mass Storage Systems, Monterey, California*, pages 101–110, 1993. IEEE Computer Society Press.

[2] M. Delfino. Rapid Access to Event Subsamples in Large Disk Files Through Random-Access Techniques. In F. Abe Y. Watabase, editor, *Proceedings of CHEP91, Computing in High Energy Physics, Tsukuba (Japan)*, pages 353–358, 1991. Universal Academy Press.

[3] C. Curran et al. The FATMEN Report. Technical Report DD/89/15, CERN, 1989.

[4] J. Shiers et al. *HEPDB - High Energy Physics Data Base (Version 0.03).* CERN, 1992.

[5] J. Altaber et al. CHEOPS – Really Using a Satellite. Technical Report CN/19/5, CERN, 1991.

[6] S.Amato, J. de Mello Neto, J. de Miranda, C.James, D.J.Summers, and S.B.Bracker. The E791 Parallel Architecture Data Acquisition System. *Nuclear Instruments and Methods in Physics Research*, A324:535–542, 1993.

[7] A.L. Peterson. E-Systems Modular Automated Storage System (EMASS) Software Functionality. In B.T. O'Lear, editor, *Proceedings of the 11th IEEE Symposium on Mass Storage Systems, Monterey, California*, pages 67–72, 1993. IEEE Computer Society Press.

[8] R.P. Mount. Data Access and Data Storage in the LHC Era. Technical report, CERN, 1993.

[9] M.W. Collins and C. E. Mexal. The Los Alamos Common File System. In B.T. O'Lear, editor, *Proceedings of the 9th IEEE Symposium on Mass Storage Systems, Monterey, California*, 1993. IEEE Computer Society Press.

[10] J. Allison. The NCAR Mass Storage System. Technical Report NCAR Scientific Computing Division, NCAR, 1988.

[11] David Tweten. Hiding Mass Storage Under Unix. In B.T. O'Lear, editor, *Proceedings of the 10th IEEE Symposium on Mass Storage Systems, Monterey, California*, pages 140–148, 1990. IEEE Computer Society Press.

[12] P. Honeyman C. Antonelli. Integrating Mass Storage and File Systems. In S. Coleman, editor, *Proceedings of the 12th IEEE Symposium on Mass Storage Systems, Monterey, California*, pages 133–138, 1993. IEEE Computer Society Press.

[13] J. Goldick, K. Benninger, W. Brown, and C. Kirby et al. An AFS based Supercomputing Environment. In S. Coleman, editor, *Proceedings of the 12th IEEE Symposium on Mass Storage Systems, Monterey, California*, pages 127–132, 1993. IEEE Computer Society Press.