The Medium is NOT the Message
OR
Indefinitely Long-Term File Storage at Leeds University

David Holdsworth
Computing Service
Leeds University
LEEDS LS2 9JT
United Kingdom
dholdsworth@leeds.ac.uk
+44-113-233-5402
+44-113-233-5411

Abstract

Approximately 3 years ago we implemented an archive file storage system which embodies experiences gained over more than 25 years of using and writing file storage systems. It is the third in-house system that we have written, and all three systems have been adopted by other institutions.

This paper discusses the requirements for long-term data storage in a university environment, and describes how our present system is designed to meet these requirements indefinitely. Particular emphasis is laid on experiences from past systems, and their influence on current system design. We also look at the influence of the IEEE-MSS standard.

We currently have the system operating in 5 UK universities. The system operates in a multi-server environment, and is currently operational with UNIX (SunOS4, Solaris2, SGI-IRIX, HP-UX), NetWare3 and NetWare4. PCs logged on to NetWare can also archive and recover files that live on their hard disks.

Background

The earlier part of our experiences has a rather UK-specific flavor. Our 1968-1972 system had a large in-house element and ran on an English Electric KDF9 system[1]. From 1972-1980 we used ICL's George3 system[2], where the two-level filestorage was part of the system. In addition to on-line files, off-line files on half-inch tape were still part of the file system.

Our procurement of a system for the 80s brought us into contact with some of the harsh realities of the file systems of the time - particularly so as the decision was to go for the then rather new VM/CMS. This led to a second in-house system - known rather unimaginatively as the Leeds Filestore, and used at the universities of Reading and Warwick in the UK, and also University College, Dublin, and the Technical University in Braunschweig in West Germany.

Other UK universities had similar experiences, and so in 1990 a self-appointed working group formulated a set of requirements, but failed to find a product that met them.

The major points were:

- Files can be sent to the archive from any of the participating file systems on the campus.
- Recovery of a file can be onto any system, not necessarily the originating system.
- The retention of indexing information is done by the system.
- It should be easy for an end-user to rename files.
- The overheads per file must be very small, as many of the files are themselves small.
- The system should be able to exploit new storage technology seamlessly.
- The system should cope with data for a shifting population of many thousands of users.
- Data should be safe.
- There should be no reliance on operating system modifications.

We at Leeds implemented a system to meet the most important of the requirements, one of which was that we wished never again to need a new system. The resulting system and the experience that led to it form the subject of this paper. Like its predecessor this system has never really been given a name, so for now we shall merely call it the LEEDS system, claiming the acronym L*eeds* E*ver-lasting* E*xtensible* D*ata* S*tore*.

The requirement is for something rather less than direct access to MSS volumes from applications programs. We need an MSS-type system into which users can consign their files for safe keeping. The actual processing of data by end-users will be in the form of files in "standard" manufacturers' operating systems. Most of the helpful concepts from IEEE-MSS are actually from version 4. The drift in version 5 seems to be more towards end-user or applications programs being aware of the existence of MSS media.

Lessons from the past

Identification of users

Organisations change on timescales which are short compared to the lifetime of data. Departments get restructured, and user-names sometimes change, either as a result or because of name changes. In the past we have had a hierarchy of user naming based on departments (Geroge3), and we have also labelled data on off-line media (tape) with user-names. We do neither of these things now, although Novell's NDS is pushing things in the direction of user hierarchy, and eroding the importance of the internal object ID. The LEEDS system uses an internal ID for each individual, and there is evidence from BrainShare[3] that Novell are also moving back in that direction.

Staging of files

With George3 we had a system in which the user need not be aware whether a file was on-line; a request to open a file that was not on-line brought it on-line transparently. This transparency is actually very visible in the time domain. Users need to be able to stage requests for their files ahead of needing them. Armstead and Prahst[4] report the same lesson. In our systems since 1980 we have gone to the point of treating staging as the norm. Files are not automatically migrated just by referring to them. Where users have interactive access to the file system this has proven not to be a problem.

Naming Systems

Not only do organisations change, but the IT equipment changes. Our previous file archives were primed with the data from their predecessors. They were of course associated with the main frame systems on which they ran. We have now got a system in which the archive is a separately identified name-space, and it records from which system each file came. Any file can be recovered onto any currently existing system. This neatly deals with access to data from systems which no longer exist.

Keeping Data Forever

When we abandonned the KDF9 for George3, we transferred only material which seemed to be useful - mainly source text of programs and cosmic ray data belonging to our physics department. Material such as assembly code (including the system itself) was discarded as useless. Some years later, the Science Museum in London looked at the possibility of recreating computing of the 60s by emulation. We were asked if we had the capability to provide the system software in machine readable form.

The total disk storage on the KDF9 was 48 Mbytes, and the total file store was about 10 times that size. The cost of keeping it for ever would now be all but zero. Of course, its value would also be zero if it were not indexed in some way. Such is the advance of storage technology that it is not cost effective to discard old data if it is held on modern media. One copy of all the data from our now discarded VM/CMS system occupies 14 volumes in our EXB-120, compared to the 1102 half-inch tapes that previously held the same data.

Our philosophy has always been to preserve the data and not the medium onto which it is written. Our ambition is for an environment where users need not feel the need to delete data just to recover disk space.

Our present system is already designed to drive multiple robots of potentially different technologies, in such a way that data migrates automatically and routinely onto new media (see *Robotics* below).

Integration with back-up

We have learnt that there are advantages in system integrity when the archive is integrated with the back-up system. This has been the case with all systems up to (but not including) the present one. We have also learnt that there is wide-spread and vehement disagreement on this issue.

Size of index information

We had experience (with George3) of a system in which the amount of index information associated with data increased as the data became older. This only became a problem after the system had been running for about 6 years. We have thus always been careful to avoid indexing by use of structures which have the capability to grow faster than linearly with the amount of data in the system.

Format of off-line media

Our transfer of data from the VM/CMS archive relied heavily on our knowledge of the data format on the tapes. (We wrote and owned the software.) Some UK universities used a bought-in system for which they were unable to obtain specifications of tape

format, leading to a hiatus at the end of the lives of the CMS systems. The moral is that when buying in archive systems, the knowledge of the format of the data on the media should be part of the contract.

Overview of the current LEEDS system

End-users choose to move their files into the archive, or to recover them from it. The archive needs to be made aware (by the system management) of the existence of the domain in which the user has an authorised user-ID. It is this authorisation which controls access to the system. There is no need for a separate end-user registration for the archive (although it could be managed in this way by an installation that had that policy).

We thus think in terms of an archive server which is aware of a number of client file domains. Each of the client file domains is itself a name-space with a number of servers. UNIX file systems are accessed using NFS, and NetWare clients are accessed using FTP. There are no modifications to the operating systems of the client file systems.

Notification of a request is by placing a small file in a key directory. There is one such directory for each client domain. The archiver machine (a SPARC-20) polls these directories at regular intervals. Although this mechanism was initially thought to be an interim, awaiting a more elegant solution, it has stood the test of time, and we now have no plans to change it. Figure 1 gives a simplified schematic diagram of the archiver's position in the Leeds University installation. There are actually many more servers and domains. The section on *System Integration* below gives more detail.

Naming and indexing

The bitfile concept of Mass Storage System Reference Model Version 4 (MSSv4)[5] introduces an abstraction which, for us, neatly highlights the media independence of data objects (such as users' files). It is at the heart of our current approach that an ordered sequence of contiguous bytes is the basic object of data retention. The bitfile-ID associated with it provides the handle by which it can be located on an appropriate storage volume, whereas other indexing activity maps a user's view of the object's name to its bitfile-ID.

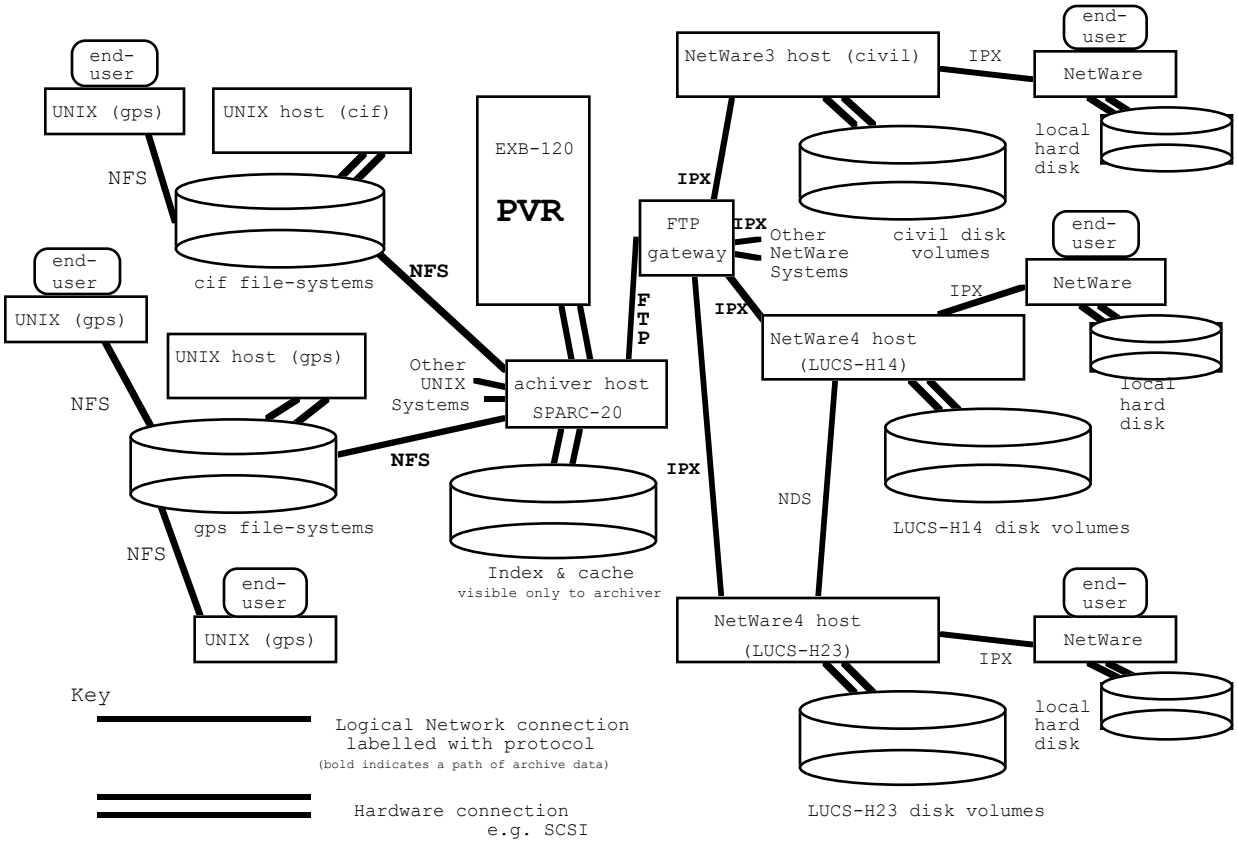

Figure 1: Schematic diagram of the archiver's position in the Leeds University installation

The naming convention of Mass Storage System Reference Model Version 5 (MSSv5)[6] is based around the all-embracing Storage Object ID (SOID). The SOID concept covers the naming of all storage objects, both physical and abstract. The SOIDs are typed, and thus the bitfile-ID concept has evolved into one particular type of SOID. The design of the LEEDS system is centred on the retention of virtual storage objects, and so continues to use the term bitfile-ID for the SOID of a Virtual Storage Object. Figure 2 shows the mapping process from end-user's name to data on a storage volume.

It seems that our preference for centrality of the bitfile concept is shared at CERN[7].

User name space

We next look at the user name space, and the mapping of file names as understood by end-users into bitifle-IDs.

The end-user sees the world in terms of user-names, each of which exists in a particular domain. Each user-name on a particular system is seen as having a filestore tree. We find that this abstraction fits well with UNIX and with NetWare, and it is clear that some other systems can also fit this model. The indexing in the LEEDS archive operates in two places. There is a mapping between bitfile-IDs and their corresponding file names as perceived by the end-user, i.e. a system-name, path-name pair. This mapping can be extracted by the end-user as a browsable file.

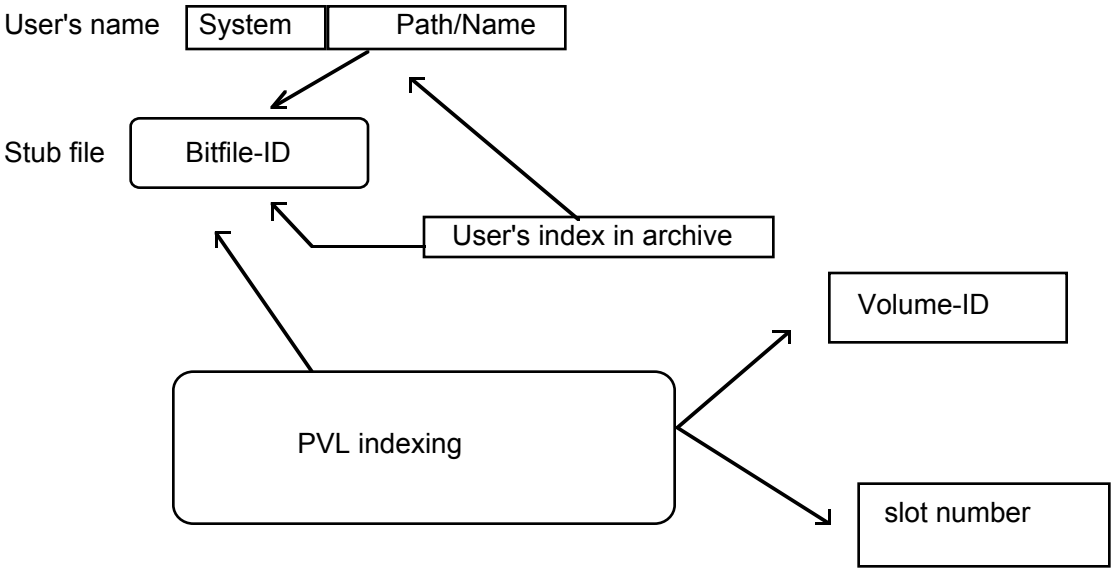

Figure 2: Name mappings

When a file has been migrated to the archive, it is replaced by a stub (of 120 bytes) which contains the bitfile-ID. This stub provides an alternative mapping between the user's view of the filename, and the real file. The stub can be copied or moved around - even between systems. It is a normal text file. It can be recreated from the indexing data held in the archive. This stub is then used as the argument to the recovery operation in order to reverse the migration process (see *End-User Operation* below).

System name space
There are 6 vital flat name-spaces. In terms of the MSSv5 model, each would appear to correspond to a particular type of SOID.

The *bitfile-ID*s are never re-used. The format is 14 letters - although the magic number 14 is in a single `#define`. This gives 26^{14} (= 6.45×10^{19}) names with the current format.

A *volume-ID* of 8 characters is assigned to each near-line or off-line volume (currently 8mm helical scan tape). Again the length is in a single `#define`.

A *media access point* (8mm tape drive in the present implementations) is named by a single letter of the alphabet. At present the limit to 26 such devices does not seem to constrain our ambitions. It is also the same as the "mount point" of MSSv5, as we have not covered the distinction between a cartridge and a volume.

A *near-line volume location* is an integer, and corresponds somewhat with the notion of slot in MSSv5. Each of these integers addresses a slingle "virtual slot" in the Physical Volume Library (PVL). Although we currently have only a single media domain, the design of the API for driving the robotics (= PVL) provides an abstraction which allows for addition of extra domains, of possibly different media types (see *Robotics* below). A volume will have no near-line volume location when it has been removed, but the system retains knowledge of its existence, and its contents.

*Internal user-ID*s are integers. In practice large institutions like universities already assign unique numbers to the individuals with whom they are involved. Each of the five user sites has been able to integrate this aspect into its existing system for registration of end-users. These integer user-IDs provide the basis for management of ownership and access permissions.

Each *client system* has a *name* which is a character string -- often its IP host name, or yp-domain name. For each client system we maintain a simple file which maps the end-users of that system to the unique user number. Thus an individual may have identities on several systems, but have them map to one common archive owner.

Robotics (Physical Volume Repository/Library)

The robot driving component of the system is driven through an abstract API in which the key system call is for the mounting of the contents of a volume location (i.e. slot) into a particular drive (mount point/media access point). The reply to a call of this API routine has four possible outcomes.

1. OK - volume is mounted.
2. Cannot do it now - should be possible later
3. This operation is not possible
4. Hardware malfunction

A vital part of this API is the possibility of the reply that the requested mount can never succeed. This allows our simple naming scheme to work with multiple robots of mixed technologies, by building an implementation of the API in which each separate robot has associated with it a subset of the drive letters, and a subset of the (PVL) slots. It also caters for a robot such as the multiple Panasonic MARC machine in which full traverse capability is not available to each robot arm.

The actual robotic hardware in use in the 5 currently operational installations is from Exabyte:

2 systems with EXB-120 and 4 x EXB8500c drives,
2 systems with EXB-480 and 3 x EXB8500c drives.
1 system with EXB-480 and 4 x EXB8500c drives.

For test purposes, we also have an implementation of the PVR for use with a human robot who just loads tapes from a shelf of numbered slots into tape drives following instructions displayed on a screen..

System Integration

Each client file domain contains software (the `rkv` command) which writes queue entries into a directory reserved for the purpose, and each UNIX file system needs to contain a special directory which then links to files which are awaiting access by the archiver.

The main archive system runs on a dedicated SPARC-20 machine, with 8 Gbyte of disk space. The bulk of the disk space is used for cacheing of files in transit. Files newly migrated to the archive reside in the cache, and a periodic dump operation writes such files to tape. A periodic cache purging operation removes files on an LRU basis.

A request to recover a file leads to its being recovered from tape into the cache, and it is transferred from there to the end-user system. In the event that the bitfile is already in the cache, the first stage is omitted.

The UNIX client file domains export their file systems to the archive machine, and so the data is manipulated directly using NFS. The NetWare clients permit read and write to their volumes. The FTP access is via a small gateway server which has a dedicated UTP ether connection to the archiver. This is the only subnet on which the archive NetWare password appears in clear.

End-User Operation

A file (or more usually several/many files) is (are) tranferred to the archive by an explicit user command (called `rkv` - chosen so as not to clash with any system's built-in commands, and yet still sounding a bit like the word "archive"). There are three possible operations on a file:

migrate - transfer the file to the archive
recover - get the file back from the archive
back-up - copy the file to the archive

The `rkv` command is available for both DOS and UNIX environments. There is also an add-on to the Windows file manager.

In addition to the three major operations, there are facilities for recreating stub files, and for obtaining directory information.

System Management

The system is designed to run with minimal attention. This is normally the case. If the network is behaving well, the main operational task is the feeding of blank tapes, and the removal of tapes to secure remote storage as a disaster precaution

There needs to be a regime for updating the maps between user identities on client domains and the internal user-ID. The detail of this depends on the site policies.

Data Integrity

The tape handling regime ensures at least 2 copies of each bitfile, with the added requirement that there must be at least 3 copies if there are no disk copies.

The bitfiles held on a tape are in order of bitfile-ID, and contain all the bitfiles in the range. This means that in order to index the total contents of the volume all we need is to

know the starting and ending bitfile-IDs. A separate index for each volume gives the block position of each bitfile. The complete set of bitfiles then resides on a set of tapes, forming a stream of bitfiles. The 3 copies of each bitfile are obtained by having three streams. We replicate the bitfiles themselves, not the volumes on which they reside.

When a file is migrated into the archive, it is written in duplicate onto different disk volumes. The periodic dump to tape extends the shortest of the 3 streams by writing onto a fresh tape, or by a complete overwrite of an obsolete tape. When this is complete one of the disk copies is deleted. Each bitfile is thus copied to tape 3 times, and only after the third time is the remaining disk copy a candidate for cache clearance. The time interval between periodic dumps is chosen so as to match the likely usage level. In the event of the caches becoming unusually full, an extra dump is run. The data is safe against filling of cache residences.

Obviously the dumping process generates a number of part full volumes. There is a copy process (for the most part initiated automatically) which copies multiple input tapes from the same stream onto a single output tape. When the output tape is full, it leaves an overlap in the stream between the output tape and the incompletely read input tape. A subsequent copy operation will then carry on from this point. The live listing of volumes in the Leeds University system can be inspected on the Web site.

An important property of the copying operation is the ability to substitute data from another stream in the event that the tape volume that would naturally be used is not available in near-line storage. One of the streams is routinely held remotely from the main machine room, but its contents can still be copied. Also, when a tape volume fails the procedure is merely to remove the offending volume and then instruct the system to copy it.

Operational Experience

There are currently 26939 user-IDs registered on 18 client domains in the system at Leeds University, which is the most mature of the 5 sites. It holds about 2.7 million files, with a total size of about 0.5 Tbyte. The system is coded to allow access to files which are not held in the robot, and offers a recovery time of about 2 minutes when the queue is empty. There is of course automatic batching of multiple requests from the same volume.

How safe is the data, given that computing hardware malfunctions from time to time? So far we have successfully recovered from all mishaps of this nature. This includes:

accidental loss of the index partition,
removal of a cache area,
accidental corruption of the master table of tape locations,
a very few tape failures - including two which actually snapped.

Assessment against design goals

Files can be sent to the archive from any of the participating file systems on the campus.
YES and it is open to add a separately managed departmental system, with its namespace distinct from the main campus facility. This is to ensure that management errors on the client system cannot compromise other people's data.

Recovery of a file can be onto any system, not necessarily the originating system.
YES but not between departmentally managed systems and centrally managed systems, for the above reasons of data security.

The retention of indexing information is done by the system.
YES

It should be easy for an end-user to rename files.
YES but this is true only for stub files. The name recorded in the archive is always the name that the file had when it was archived.

The overheads per file must be very small, as many of the files are themselves small.
YES - approx. 60 bytes + pathname of the file

The system should be able to exploit new storage technology seamlessly.
YES but not well tested. The implementation of human robot and of EXB-480 went smoothly. There is a present assumption of reading and writing of blocks on the storage medium via the UNIX driver, and also of FSR (forward skip).

The system should cope with data for a shifting population of many thousands of users.
YES - there are currently 26939 user-IDs registered on 18 client systems.

Data should be safe.
YES - replication and recovery techniques have been used in live situations

There should be no reliance on operating system modifications.
YES - The archiver machine in vanilla Solaris2, and all the client systems are also unmodified.

Assessment against IEEE-MSS standard

Our two major omissions are in relation to virtual storage objects, and PVR cartridges.

Firstly, our bitfiles are constrained to be a contiguous sequence of bytes, not the multi-segmented virtual storage objects of MSSv5. However, in defence we would argue that any structure can be mapped onto a contiguous sequence of bytes, and so why stop there. One could offer the whole panoply of indexed sequential access - but it would be a mistake.

Secondly, our volumes are not housed in cartridges. Each volume is only one mountable object. This will make the driving of some types of device rather contorted, but not impossible.

Because the design of the API for access to near-line volumes was designed before the release of MSSv5, the correspondence to the PVL/PVR structure is not quite total. There is an identifiable part of our system which is the *locator* and identifies the volume-ID and its slot number. This slot number is then presented to a mount request which is best thought of as a request to the PVL. If the drive and slot number in this request are in the same PVR, this request will succeed (hardware malfunction permitting). If the drive and slot number are not in the same PVR the reply is such as to cause the system to try other drives until the operation succeeds.

Conclusion

We do not have peta-bytes of data, but we have quite a lot, and it goes back in time. We do have a large floating population of users, and the lapse of time means that systems come and go. Our techniques enable meaningful long-term data storage, and we have a thoroughly operational system running on 5 sites.

Web Site

The WWW site gives a potted system description, more historical information, and some access to live information on the Leeds University installation. The URL is:

`http://www.leeds.ac.uk/ucs/systems/archive`

References

[1] M Wells, D Holdsworth, AP McCann, “The Eldon2 Operating System for KDF9”, *Computer Journal* vol14 no1 (1971)

[2] G.B.Newell, “George 3, The Compleat Operating System”, J.Inst. Comput. Sci. vol3 no3 (1972) *also* International Computers Ltd, “Operating Systems George 3 and 4”, Technical Publication 4267, 4th edition (1971)

[3] Dave Eckert, “NetWare Directory Services (NDS) Futures”, DSS201T *BrainShare ’96*, Novell Inc 1996

[4] Betty Jo Armstead, Stephen Prahst, “Implementation of a Campus-wide Distributed Mass Storage Service: The Dream vs. Reality” *Fourteenth IEEE Symposium on Mass Storage Systems* IEEE 1995

[5] Sam Coleman, Steve Miller, Eds., “Mass Storage System Reference Model: Version 4” IEEE draft 1991

[6] Lester Buck, Sam Coleman, Rich Garrison, Dave Isaac, Eds., “Reference Model for Open Storage Systems Interconnection” Project 1244 - IEEE 1994

[7] Jamie D Shiers, “Data Management at CERN: Current Status and Future Trends” *Fourteenth IEEE Symposium on Mass Storage Systems* IEEE 1995