Multi-Level Storage: a User's Tale

Andrew Findlay

Brunel University
Uxbridge
GB
Andrew.Findlay@brunel.ac.uk

ABSTRACT

Brunel University installed a multi-level filestore in 1992 to serve a user population of 7500 staff and students. A number of unexpected problems occurred which were mostly attributable to the particular demands of the large student population. The students have now been taken off the multi-level store again, but some of the complications remain.

1. Background

November 1991. Brunel has 5GB of user filestore spread across ten small filesystems. Every filesystem is over 90% full. We have one systems administrator working full time on shuffling users from disk to disk and compressing files to keep some space free. The largest disks available are 1GB and it costs about POUNDS3000 per GB to build fileservers. There are 4000 users and two systems administrators. Filestore requirement is doubling every year We have to reduce the effort spent on managing the filestore.

2. Optical Storage

Optical disks had been available for some time, and the rewritable ones were starting to appear in serious products. Jukeboxes holding several drives and a number of disk cartridges were also appearing on the market.

Our thoughts turned to automounters controlling jukeboxes, but the spectre of managing *hundreds* of 300MB filesystems was horrifying. There would undoubtedly be performance problems as well, with people demanding files from many disks simultaneously. There had to be a better approach.

3. Multi-Level Storage

Other institutions were having similar problems, and by comparing notes we found the names of some "Multi-Level Storage" (MLS) products. There were not many that would work with Unix systems, and of these, only Epoch looked credible.

The ideas are not new: ICL's George III operating system used Multi-Level Storage in the 1970s, but using optical disks and jukeboxes rather than magnetic tape and operators promised great performance gains over the systems of old.

Multi-Level Storage is based on the observation that most files are not used for months on end. If files not likely to be used in the near future can be moved to a slower cheaper form of storage, better use can be made of the expensive fast disks. The forecasting tends to be based on statistics: the longer is is since a file was last used, the longer it is likely to be before that file is used again. Unused files migrate out to jukebox stores, and get staged-in again when they are accessed.

We looked at our filestore and found that over 70% of the space used belonged to files that had not been used for two weeks or more. Figure 1 shows the age/size distribution for some sample filesystems.

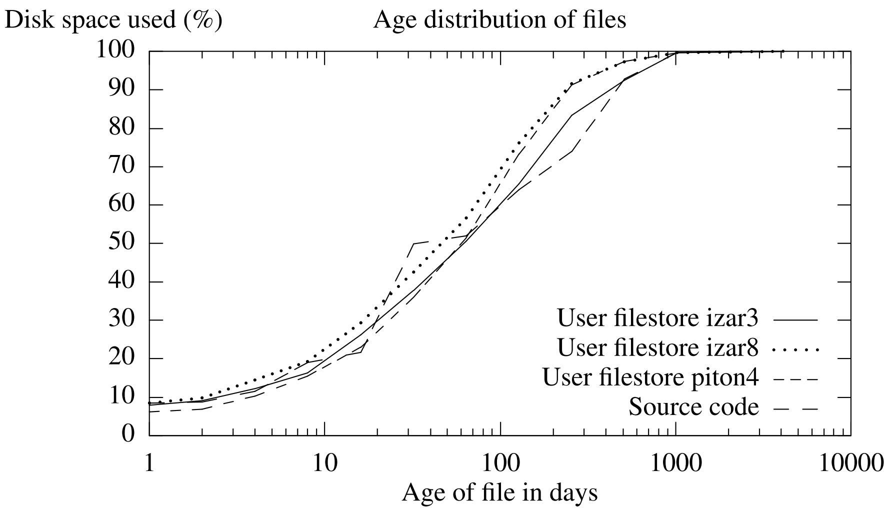

Figure 1: Space used vs time since last access

We looked at the economics: an Epoch system was cheaper than conventional fileservers where more than 40GB of nominal capacity was needed (Figure 2). A 60GB system would cost about POUNDS100k including the migration software for our existing fileservers.

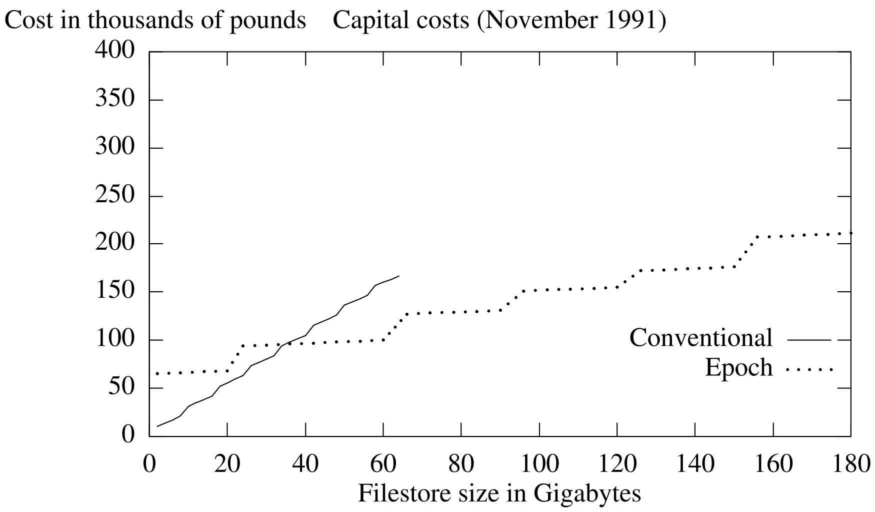

Figure 2: Capital costs of fileservers in 1991

We looked at backup: Epoch's baseline backup would cut the backup time dramatically and possibly also provide faster disaster recovery than the conventional level-zero-on-tape scheme we had been using. It was more complex, but had tape management features which would help with user-requested restores.

4. Purchase

We had just enough money available as a result of Computer Board's last-ever capital grant, so we bought a 60GB Epoch-1 with migration software for our two main fileservers. The deal also provided for upgrading to a new Epoch-2 when they became available. The machine was installed in March 1992 and tests were completed just in time to enable migration on two hard-pressed filesystems before the end-of-term overflow. Congratulations all round.

5. Phase-in

During the summer term we gradually moved more files onto "migrating" filesystems. The plan was to have three 1GB filesystems on the main fileservers acting as the front-end, migrating onto a 1GB cache disk on the Epoch and then onto optical disks in the jukebox (Figure 3). Each filesystem was to be built with one million inodes to allow for two years growth in the number of files. We expected the front-end filesystems to hold all files that had been used in the past month, with the cache disk holding another week's data.

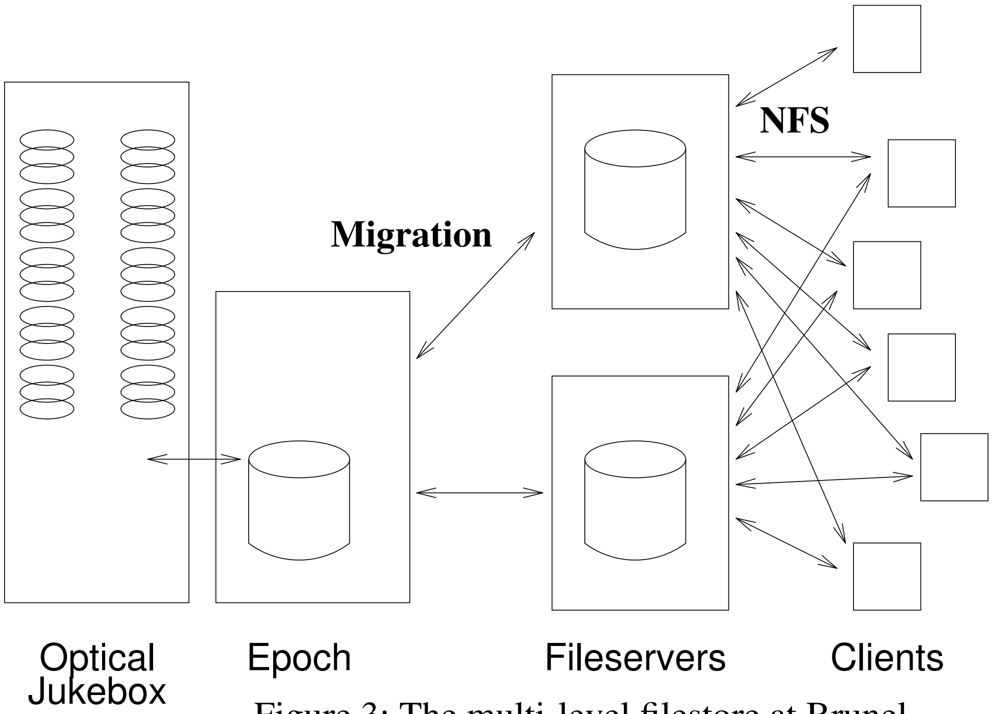

Figure 3: The multi-level filestore at Brunel

Wrong! We had not allowed for the space used by a million inodes, nor for the space used by a million records of "extended inode information". After subtracting these two plus the normal 10% free-space margin we ended up with 600MB available on each disk. It is necessary to run migrating filesystems at about 90% of maximum capacity to allow for growth during the day, so there went another 60MB. The remaining 540MB should still have been enough for two weeks file turnover, but the migration statistics showed just eight days on two of the filesystems. As a lot of activity in a university has a seven-day cycle, we were most concerned to keep more than seven days-worth of files on fast disks.

Storm clouds were already gathering: a faulty picker-arm in the jukebox caused trouble, and both optical drives had to be replaced too. There were software problems as well, particularly with the Mount Queue Manager which controls the jukebox and keeps track of tape volumes.

Before the students arrived for the new academic year, we doubled the number of disks used as front-end cache and arranged to stage-in all the "dot files" and other critical things so that logins would not be delayed by jukebox activity.

6. NFS Server izar not responding...

We had an awful term. The jukebox activity increased to over 10000 movements per week, and the number of soft errors rose with it. The suppliers said they had never seen a jukebox work so hard. The high level of activity exposed a lot of problems, particularly in the Mount Queue Manager which tended to regard everything as a fatal error and either die completely or set off to make an inventory of the jukebox contents. As the inventory process takes 20 minutes and no stage-in requests can be serviced until it is finished, this was a major showstopper.

The really nasty problem with migration failures is one that could apply to any slow filesystem exported by NFS. It happens like this:

> Someone tries to read an old file
> Their workstation sends an NFS request to the server
> The server sees that the file needs staging in, and requests it from the jukebox
> The NFS daemon waits for the data
> If the data becomes available quickly then all is well, but if not...
> The workstation times out and sends another NFS request
> Another NFS daemon on the server requests the file from the jukebox...
> And so on until all NFS daemons are waiting for the same file

It is therefore possible for a single user process to jam an entire fileserver by requesting a file that cannot be staged-in. CDROM users beware!

We developed workarounds: scripts to restart the Mount Queue Manager if it died, and scripts to watch for NFS daemons getting locked up. It was possible to maintain a rather lumpy service even with the jukebox completely offline by killing and restarting all the NFS daemons every minute or so. Unfortunately the kernel objects to too much of this treatment and after an hour or two the only solution was to reboot the fileservers.

Eventually, Epoch fixed the worst MQM problems and found a way to make NFS daemons time out. January 1993 was almost pleasant: the incident rate was down to under one per week at last, but there was another problem waiting.

Our users had discovered graphical file managers.

7. Filemanager vs MLS

The OpenWindows file manager is an important part of the GUI environment. It displays each file as an appropriate icon and will invoke the right application when the user selects a file to work on. A library known as the "Classing Engine" deals with classification of files and selection of icons. It works by matching the filename, permissions, and contents against a set of patterns to decide what application the file belongs to. As the file manager invokes the classing engine on every file it is going to display this means that the first few bytes of every file need to be read whenever filemanager enters a directory.

Filemanager can wreak havoc with a multi-level filestore. It keeps reading files that are never used, so the multi-level store thinks that they are in active use. Worse still, the first time a user tries filemanager every migrated file in their home directory gets staged-in. *One at a time, and in alphabetical order.* The files would have been staged-out in some order based on size and time since last access, so each one might reside on a different optical disk. This treatment was driving the jukebox robot to an early scrapyard.

Filemanager was winning so we disabled it. This was not popular, but fortunately Sun were aware of the problem and we soon had a patch to prevent filemanager from reading files at all. Most of the pretty icons and automatic application-launching vanished, but at least we could declare a draw and allow filemanager to be used again. The real solution is probably some sort of lazy-evaluation switch so that icons depend only on filename and permissions but full classing is done if the user actually selects a file to work on it.

8. 20GB of User Data > 60GB of capacity!

Another problem appeared: the system was eating optical disks at a ferocious rate. With these costing POUNDS100 each at the time the cost alone was a problem, but the real worry was that our "60GB" jukebox was almost full and we only had 20GB of user data in total! A hunt was convened to find the missing space.

A quick check revealed 10GB of incremental backup files belonging to the Physics Department's VMS cluster. It appeared that they were not deleting expired backup files, but a quiet word with the system manager soon produced results and we had 8GB back. That left about 30GB to account for.

A major user of space is the Epoch backup system. It maintains catalogues of each dump so that files for retrieval can be located automatically and the correct tape can be requested. At the time we were trying to run two backups each week: one for offsite storage and one for local uses such as user-requested retrievals. The local backups must be kept for a year to match the main fileserver backups, so their catalogue entries must be kept as well. This was using 10GB but we could hardly consider it to be "wasted space".

What of the remaining 20GB? Well, it simply did not exist: it was "stale space" on the optical disks. This is a normal feature of optical filesystems but we had not expected there to be so much. To understand how this happened it is necessary to know something about magneto-optical disk technology.
The important facts are:

- Optical disks are relatively slow. They can read about 350kB/second, where magnetic disks manage about 1.5MB/second.
- It takes three rotations to write anything on an optical disk. First the area to be written must be erased, then the data is written, then it is verified. Thus write performance is much lower than read performance.

The Epoch does several things to improve performance:

- Complete disks are pre-erased
- As far as possible, all writing to optical disks is done in overnight batches.
- Files are written as contiguous chunks
- All optical disks are treated like WORM (Write Once Read Mostly) media: The space belonging to deleted files is not re-used so disks accumulate stale space and must eventually be compacted.

Our problem was caused by compaction (or rather, a lack of compaction). The Epoch can be set to keep a certain number of optical volumes available by compacting partly stale volumes automatically. The effort involved in compaction is related to the amount of stale space on the disks. If disks average 10% stale at the time of compaction, ten disks must be re-written to free one complete disk. Allowing disks to become 20% stale before compaction reduces this number to five. There is an obvious trade-off between efficiency of media use and time spent moving data from one disk to another.

The human element steps in at this point in the story. Our operations staff had been instructed to make sure that there was always at least one complete disk free, by feeding in new optical disks whenever necessary. The Epoch itself was aiming for five *sides* free, which it could achieve very easily because it kept being given new blank disks! When we realised what was going on and changed the parameters we found that the normal three hours of compaction activity each night was not enough to make headway, and had to schedule 17 hours per day for several weeks. Part of the problem was that compaction is not the only process that needs access to optical disks (See Table 1) and all processes must compete for the use of the two drives. Another problem was our higher-then-expected file turnover, which was causing disks to go stale very rapidly.

- Stage-in (mostly user-driven and thus random)
- Stage-out (mostly scheduled in overnight batches)
- Compaction and erasing (scheduled overnight)
- Backup (scheduled, but usually runs for several days)
- Recover (if baseline volumes are needed. Done at user request)

Table 1: Processes that need access to optical disk drives

9. Backup and Recover

Backup is an extremely important chore. In most systems the process is easy to understand and the decisions to be made tend to be "How Often?" "Will we use incremental dumps?" and "Which tapes do we send off-site?". Recovery is usually simple if the backup procedure has been set up carefully.

Multi-level stores add some interesting complications. There are three main reasons for this:

- Files migrate between different storage media, but backup coverage must be complete.
- MLS systems tend to be too big for conventional level-zero dumps to be practical.
- Restoring a level-zero after a disaster could mean writing hundreds of optical disks, which would take weeks.

The Epoch solves the second problem by introducing a "baseline" backup. This is a "level minus-one" backup which is written to optical disks *in the same format as the working media.* It contains "stable" data: files that have been migrated to optical media and have not been changed for some time. The idea is that in a disaster-recovery situation these disks are simply loaded into a new jukebox and become the new working media.

Level-zero dumps can now avoid dumping files that are already on a baseline disk. In effect the level-zero becomes a form of incremental dump, but it always includes all the filename and inode information.

In a migration setup like ours, the fileservers "fronting" the Epoch must be backed up as well. They have all the filename information† as well as all the files that have not migrated out to the Epoch. To ensure full backup coverage, the fileservers must also back up any files that have been migrated out but have not yet been backed up on the Epoch itself. This final requirement can cause performance problems, as the fileserver backup program must ask the Epoch about the status of every migrated file. Worse still, if the fileserver finds itself backing up files that have migrated right out to optical disks it can cause severe thrashing of the jukebox.

We expected that files would be fairly stable when they got as far as the optical jukebox, so we set the baseline backup to copy files that had been on optical disk for two weeks or more. We soon found that the baseline sets were growing rapidly, so we extended the interval. As a result, each level-zero dump used more tape. The original plan had been to keep level-zero dumps below 5GB so they would fit on a single Exabyte cartridge. Tape is relatively cheap, but with backups now requiring several tapes and taking days to complete we had to buy a tape jukebox to keep the process moving overnight. The level-zero dumps seem to have stabilised now at four tapes and they take about 40 hours because a lot of data has to be read from optical disks.

Optical baselines become stale just like the working disks, so they need compacting from time to time. The catch is that these are *backups:* they exist to allow deleted files to be recovered, so all that "stale" space actually contains important data! In practice this means that compacted baseline volumes cannot be re-used until *all* tapes that refer to them have expired. In our case this means that baseline volumes do not become free for a year after they have been compacted. Where file turnover is high this can get expensive.

File recovery is a multi-stage process. Users often ask for whole subdirectories to be restored, containing files with a wide range of creation dates. Some of these files were probably on the Epoch disks or in the optical jukebox when the dump was done, so three sets of media are involved:

- The fileserver dump tapes
- The Epoch dump tapes
- The Epoch baseline disks

The restore process on the fileserver is fairly standard, and for recently-created files is often all that is needed. However, any file that had been migrated to the Epoch at the time of the dump will not be on the tape, so the restore program must issue an "undelete" request to the Epoch. When the fileserver restore has finished, all files appear to exist but some will return I/O errors if the user attempts to read them. The next stage is to locate the appropriate Epoch backup and recover the files that were flagged for "undelete". This is a fairly automatic process, but it does require some guidance to avoid searching every backup catalogue on the system. Once the tapes have been read, a lengthy scan process is invoked to find any files that must be read from baseline volumes.

There are a number of pitfalls in this process, of which the worst is that the system will not say in advance which volumes it wants to use so an operator has to keep inspecting the Mount Queue Manager display.

---

† When files migrate from the front-end fileserver to the Epoch they are given a file-ID number. It is this number that is used as the name of the migrated file in the Epoch filesystem.

10. Understanding the performance problems

By May 1993 we had 7500 registered users with 16GB of user-filestore. All of this was on migrating filesystems and the Epoch was working very hard. Performance and reliability were nowhere near good enough.

After many discussions with the suppliers and with Epoch's own engineers, we decided that our main problem was the very fast turnover of files. The analysis tools we had used to estimate filesystem activity work on static data: they take a snapshot of the age and size of every file at a particular instant and calculate the activity from that. Such a tool cannot take note of files that were used for a short time and then deleted, so at best it gives an approximation to the real situation. Epoch's advice was to add more cache disk and to associate particular sets of optical disks with individual filesystems on the servers to reduce the number of disks that any one user's files could be spread over. As we already had six gigabytes of disk on the fileservers and three gigabytes of cache disk to service a virtual filestore of 20GB we were not keen to add more magnetic disks. We did reorganise the use of optical media though, and added two more optical drives to reduce contention.

It seems likely that the real problem stems from our pattern of use. 20GB of filestore shared by 7500 users will have a very different access pattern from 20GB shared by 50 users. The random element introduced by this large user population seems to defeat the simple stage-out algorithm used by the Epoch software. If we had more control over the placement of staged-out files we might be able to reduce the movement rate in the jukebox, but this would be at the expense of keeping free space on almost every optical disk.

11. Backing out

As the University planned to expand continuously over the next few years the problems could only get worse, so we decided to take all taught-course students off the migrating filestore. Another market survey was done and it is interesting to note how the economics have changed. Figure 4 shows the new comparison between Epoch systems and conventional small fileservers, while Figure 5 shows historical data. When comparing Figure 4 with Figure 2 it is important to note that the later graph accounts for the overhead space while the earlier one is based on "raw" capacities.

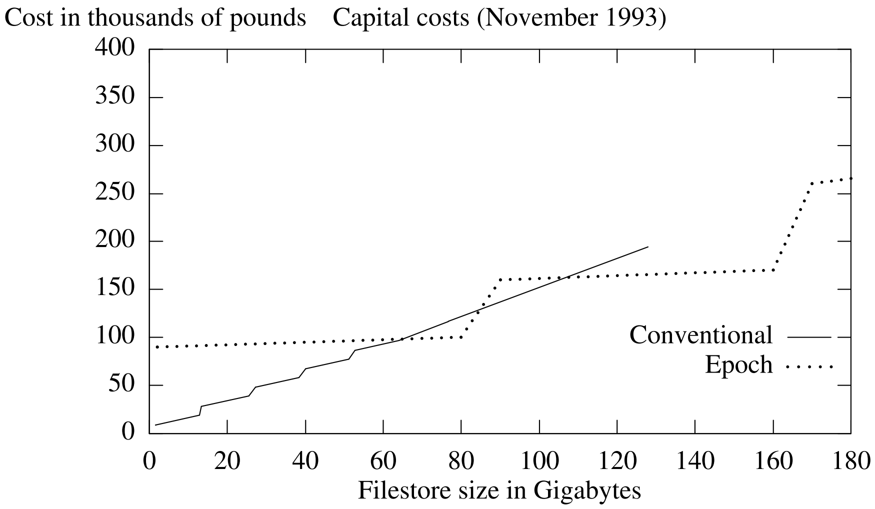

Figure 4: Capital costs of fileservers in 1993

The cost of building "conventional" fileservers seems to be falling at about 40% each year. In 1985 the most cost-effective fileservers were VAX 750s with 340MB Eagle disks, but now the best approach seems to be to use small desktop workstations with 2GB or 3.5GB SCSI disks.

A low-risk solution was essential, with no single points of failure that could affect the whole campus. We decided to use three refurbished SPARCstation-2 base units with four Wren-9 1.9GB SCSI disks on each, as this gave the lowest cost per GB while spreading the load across several servers. Each server exports

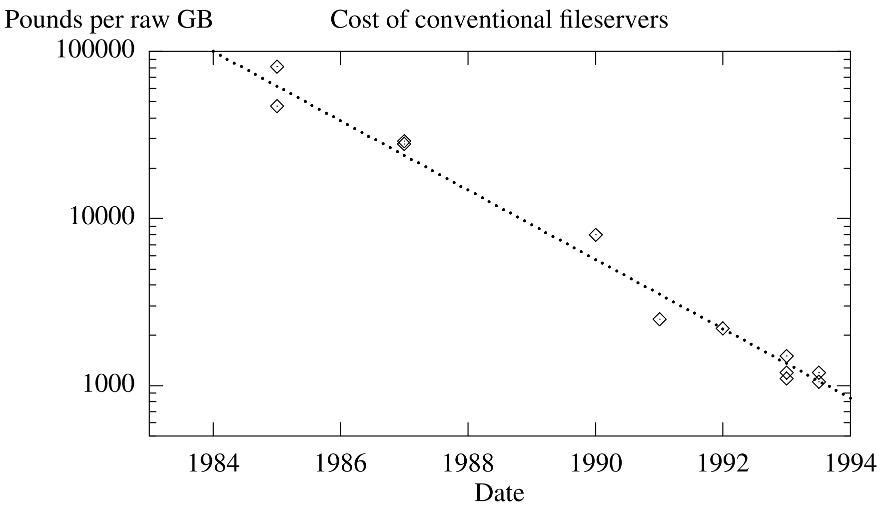

Figure 5: Cost of building conventional fileservers

about 6.5GB of filestore.

During the summer, all files belonging to taught-course students were moved onto the new servers. Groups of users were allocated to filesystems with the aim of starting the new academic year with each filesystem about 45% full. The idea is to avoid having to reorganise the filesystems until summer 1994, though we may have to do some fine-tuning during the spring vacation.

Moving files from a migrating filestore must be done with care, to avoid thrashing the jukebox. There are no standard tools for bulk-staging files on migrating filesystems, so a set of shell scripts was written to gather up the IDs of migrated files and invoke a bulk-stage-in on the Epoch.

By November 1993 the registered user population had risen to 9200, but with only staff and researchers using the migrating stores the activity rate was more reasonable.

12. Conclusions

Multi-level filestores are a lot more complex than conventional disk filesystems. They are very good at some jobs, particularly where data is being collected over a long period of time and is hardly ever modified or deleted. Applications with predictable file turnover can probably be handled too: our use for incremental dumps seems to work well.

As with many things in the past, we have proved that the university environment is poorly understood by manufacturers and that the demands placed on systems by classes of students can be far greater than those generated by "business users". It is very unlikely that we will consider putting students onto any sort of multi-level filestore for many years to come.