DMF, MLS and an NFS-Based Station MSP

Margaret S. Douglas and *Michael L. Green*, Naval Surface Warfare Center, Dahlgren Division, Dahlgren, Virginia

ABSTRACT: *At the Naval Surface Warfare Center, Dahlgren Division (NSWCDD), disk space problems are not caused by huge files, but rather by the number of files and the size of the DD60 allocation unit. We have implemented archiving using DMF and an NFS-based station MSP, with archived files residing on disks on a Silicon Graphics Power Challenge R4000. This paper presents an overview of the design of the NFS file systems and station MSP functions, problems encountered using DMF, and the solutions to those problems. This work was done at UNICOS 6.1.6 running MLS (with levels!) and has been converted to UNICOS 8.0.2.3 (PRIV_TFM and PALs), DMF 2.1.*

Background

When assembling the RFP for our current Scientific and Engineering Computing Systems at NSWC-DD, we knew from our previous 20 years experience on Control Data systems that we would need the ability to archive user files. We used this data to formulate the following requirements for what we termed a Fixed Archival System (FAS):

- retrieval of archived files must be transparent to users
- capacity of 55 Gbytes (expandable to 200 Gbytes)
- ability to perform 400 accesses/hour (expandable to 800)
- maximum retrieval time (time from user request to availability of data) = data transfer time (at an effective rate of 1.25 MBytes/second) + maximum FAS access time of 15 seconds (e.g., for a tape based archival system, the time to locate and mount the tape, and position the tape at the desired file).
- the security of files must be maintained

After a couple of false starts (a MASSTOR system that still didn't exist at the time the contract was let and an optical disk jukebox that failed to meet the retrieval time requirement when actually put to the test) an SGI Power Challenge R4000 (Crimson), with 100 Gbytes of available disk storage (expandable to 200 GBytes) was proposed.

Our CRI system is a Y-MP2E/116 with 16 DD60s and 8 RD62s. A little more than half of the DD60 space is devoted to two DMF-managed user file systems, about 500,000 files. Having upgraded from UNICOS 6.1.6 last December, we are currently running UNICOS 8.0.2.3 with MLS enabled (PRIV_TFM and PALs), DMF 2.1. Our users actively use both the security level and access control list (acl) features of MLS.

The network between the Cray and the FAS consists of two NSC DX routers each having a PB130 Host interface for the Cray, a PCDAS-1 Dual Attach FDDI interface for the FAS and a PCET5-4 network interface providing four IEEE 802.3/Ethernet connections. The Cray has two low-speed channels, each connected to a NSC router. The FAS has a FDDIXpress Dual Attach FDDI interface connected between the two NSC routers. User access to the Cray is through Ethernet connections with the NSC routers.

Knowing that archiving would not be necessary immediately, there was time to study our file size distribution as users migrated their projects to a UNIX environment. While our file distribution had always been dominated by smaller files, this became even more pronounced as individual source code and object modules became separate files.

Unlike many Cray disks which have an allocation unit (AU) of one 4K block, the allocation unit of the DD60 is four 4K blocks. We find it simpler to consider file sizes in terms of DD60 AUs than in multiples of 4K blocks. Using a size unit of the DD60 AU, our current file distribution is shown in Figure 1. Fewer than one percent of our files are one MByte (61 AU) and larger; these account for fifty percent of our space. Eighty percent of our files occupy one AU; these account for twenty percent of our space. In Figure 1, the distribution of one AU files is divided into "quarter AU" (4K) units to demonstrate how much of the space allocated to these files is actually unused. If our allocation unit were 4K, two-thirds of the space currently used by our small files would be available—this is roughly the space on an entire DD60!

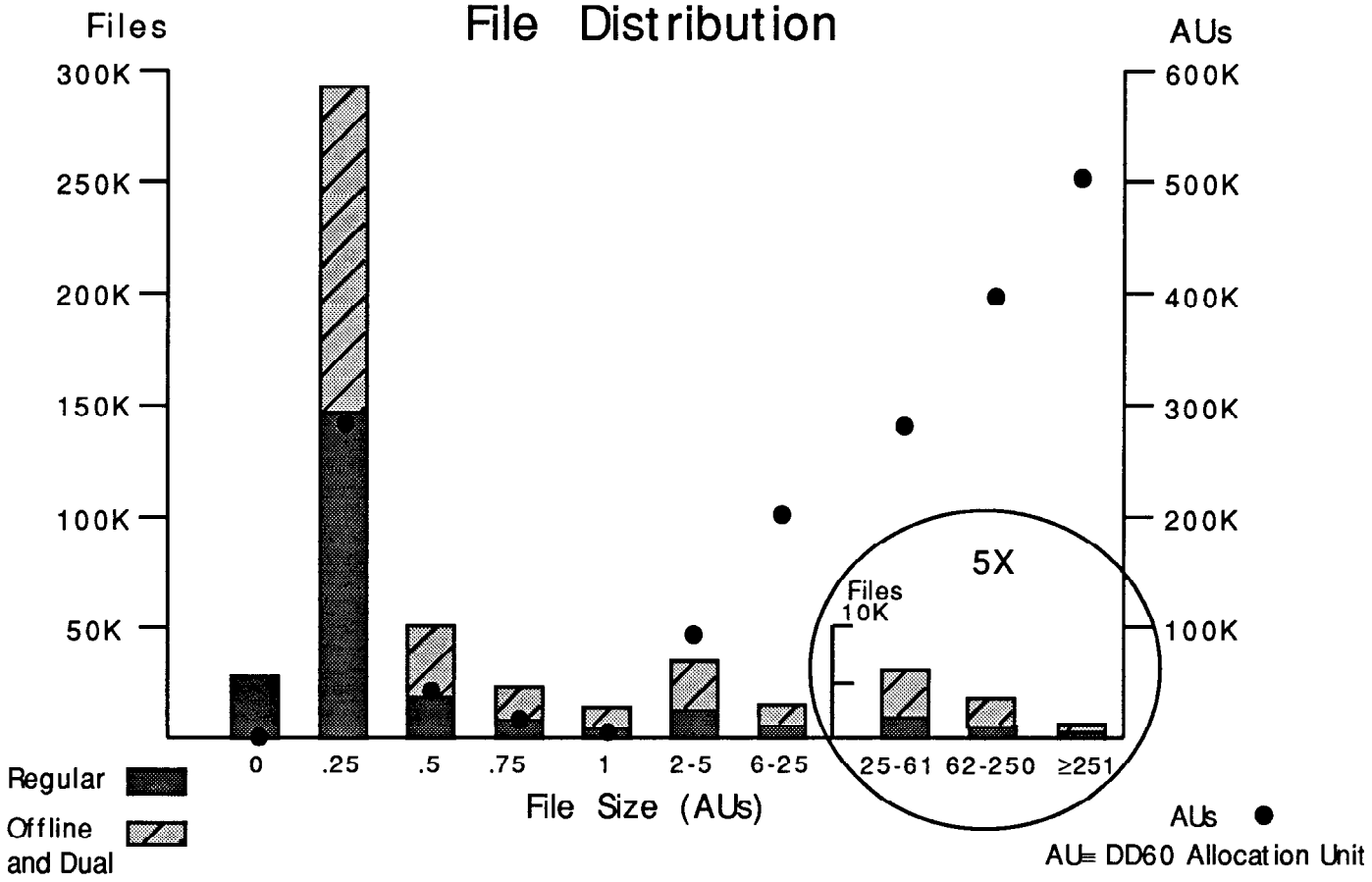

Figure 1

Approach

Our approach to the task of archiving was to determine a theory of operations (when to archive and what to archive), then configure the FAS, and use Cray's DMF, tailoring the generic station MSP to meet our needs for a reliable, responsive, and secure archival system.

When to archive

Having read SPRs and "discussions" on the unicos-l mailing list, we learned that it is not feasible to wait until we need the space to archive. We planned to archive a sufficient amount on weekends, when our system is rarely busy, and, ensuring that newly archived files have been backed up, free space daily as necessary.

What to archive

While it might seem that giving precedence in archiving to larger files would be more effective, that is not the case at our site. Given our preponderance of small files, archiving primarily large files would simply result in disks full of small files. In time, it would be necessary to archive them also. Archiving only small files is a particularly slow process; it is more efficient to archive files of all sizes at the same time. We also choose to archive small files because we are aware of how much space they "waste" due to the size of the DD60 AU. We use the default DMF weighting matrix, i.e., files are archived based solely on time of last access.

FAS configuration

In configuring the FAS, there were two major decisions to be made:

- operating system IRIX or Trusted IRIX (TIRIX)
- file system design.

The choice of operating system was an easy one. When the FAS was installed (Spring 1993), it was determined that there was a compatibility problem with the FDDIXpress software and TIRIX 4.0.4, the version available at that time. Transfer rates on the network were unacceptable, and the next release of TIRIX wasn't expected for at least eight months. Since TIRIX and UNICOS have implemented network security in different manners, there was no compelling reason to wait for the new version of TIRIX--the decision was made to use IRIX (4.0.5), and maintain security through other mechanisms.

File system design

Of primary concern in designing the FAS file systems was the ability to backup archived data. We are required to maintain

backups of all user files. The UNICOS backup (dump) utility dumps only the inodes of archived files; we "recycle" UNICOS backup tapes every four weeks. Once a file has been archived for a month, it's data is no longer retrievable from (UNICOS) tape. It is, therefore, imperative that the FAS itself can be backed up, and restored when necessary, in a timely manner.

The FAS is backed up to 8mm tape. To backup 100 GBytes would take 3 days; to restore it, even longer! Clearly, it wasn't feasible to have very large file systems, or to archive in a manner that would necessitate backing up many file systems every time. Our decision was to size file systems such that a file system could be backed up to a single 8mm tape (approximately 4.8 GBytes). File systems were striped to enhance performance. It was also decided that, as long as it continued to be feasible, in a weekly session we would archive to a single file system (the one with the most available space at the end of the previous weeks session). This way we have only to back up one FAS file system after archiving. (To avoid backing up file systems simply to reflect changes caused by hard-deletes, we maintain, for each file system, a record of all files hard-deleted. This record is kept until the file system is backed up again, i.e., the next time it is archived to.)

MSP design

When this task began, two of us studied DMF in an attempt to get a thorough understanding of how the process worked. We then gave our colleagues an overview in which we often referred to a simple diagram showing how all the "parts" (dmdaemon, UNICOS kernel, MSP, etc.) worked together. In this diagram, the MSP and the FAS were connected by a lightning bolt labelled MAGIC. We now had to define this "magic".

The most straightforward, high level, network protocol, therefore the simplest to implement, is NFS, but there were concerns about its speed. Considering, however, that the vast majority of our files are very small, transfer time becomes less significant. We chose to use NFS for our MSP transfer mechanism. FAS file systems are exported to the Cray and utilities were developed to provide the necessary MSP functionality:

"put": copy file from the premigration directory to the FAS

"get": copy file from the FAS to the unmigration directory

"delete": remove a file from the FAS.

Knowing that having very large directories has a serious adverse effect on file access time, we have attempted to minimize FAS directory search time by distributing files into subdirectories based on the Cray file's ownership. These "owner subdirectories" are created in the FAS file systems as needed.

In order to copy files of different security levels to these NFS-mounted, "foreign" file systems, a version of cp was made "trusted" in the PRIV_TFM sense. (We still use the PRIV_TFM trust definition for this, but will change to PALs in the future.) Similarly, a trusted version of mkdir is used to create the "owner subdirectories." Security is maintained as follows:

- "trusted cp" and "trusted mkdir" are inaccessible by users
- UNICOS maintains security for DMF-related file accesses
- the FAS is defined in the NAL as MAXLEVEL
- the directory on the Cray that holds the mount points for the FAS file systems cannot be traversed by users
- "normal users" do not have login accounts on the FAS
- all archived files on the FAS are owned by root with 600 permissions.
- basic NFS security checks (e.g., file systems are exported only to the Cray)
- the Cray and the FAS are on an isolated, secure network

The MSP generates Dataset Names (DSNs) of the form:

```
FAS.fs/owner/handle.checksum
```

where:

FAS.fs is the FAS file system being archived to

owner is the owner (on the Cray) of the archived file

handle is the DMF handle for the file (netaddress.seqno)

checksum is the file's checksum (used to verify the file transfer in both directions).

Use of the dmdaemon supplied file handle in the FAS file basenames ensures uniqueness.

Given that our operational plan involves weekly archiving to a single FAS file system, there had to be a way to stop the process if insufficient space remained on the target file system. The MSP utility which copies files to the FAS first determines if there is sufficient space for the file. If not, or if the file system is inaccessible, a "lock file" is created and no further transfers are attempted. Some transfers, already in progress (we have the default 8 streams running), may also fail. A message indicating the problem is issued to the operator.

Originally we did not plan to modify DMF source code except in the generally accepted areas of the generic station MSP, but problems (discussed below) required changes to dmmctl. Taking advantage of this "opportunity," we added a check for the lock file created by the MSP: if the lock file is detected, dmmctl terminates, issuing no further requests.

Daemon startup

When the system is booted, dmdaemon is not started until the NFS mounts of the FAS file systems have been verified. In order to minimize failed retrieval requests, NQS queues are not started until dmdaemon is running. We accomplish these tasks in /etc/rc.pst rather than using the "normal" sdaemon method.

Problems

Our initial operational attempt at archiving was in March 1994--we attempted to archive approximately 20000 files. The dmdaemon choked! The daemon seemed to "think" that the MSP was lost or "out of control" and would terminate and

restart it, reissuing thousands of requests. After several restarts, the daemon would terminate itself. (The MSP appeared to be functioning properly.)

We solved this problem by, effectively, slowing down dmmctl. We modified dmmctl to issue 64 requests, then wait. When final replies had been received for all but 8 of the 64 requests, 64 more were issued. dmmctl terminates when final replies have been received for all requests, or 600 seconds have elapsed since the last reply.

The next weekend, armed with our new dmmctl, we again attempted to archive a large number of files (we had succeeded in archiving about 7000 the previous weekend). This time we were too successful. The migration threshold had been reached, but dmmctl continued. When the migration threshold had been exceeded by 30000 blocks, we terminated dmmctl.

This problem was a side effect of the allocation unit of the DD60. dmmctl used a block size of 4096 bytes and the size field from the file's inode to compute the number of blocks allocated to the file. Since DD60s are allocated in units of four 4096 byte blocks, this, generally, underestimated the space used. We modified dmmctl to use the blocks field from the file's inode. This approach is not entirely correct, either, since the blocks value in the inode includes space allocated to acl's (which remain disk resident), but the number of files with acl's is small.

We've had few archiving problems since these changes. We had no difficulty porting our DMF mods from UNICOS 6.1.6 to DMF 2.1. Transitioning some of our operational procedures, e.g., gathering information on hard deleted files, log maintenance, etc., was not as simple, but no insurmountable problems were encountered.

Summary

We feel that our MSP and FAS design, coupled with standard DMF maintenance procedures, provide an archiving scheme that is maintainable, reliable, responsive, and secure.