Paper Number 2075

DCE/DFS at Northwestern University

Bruce Foster
Northwestern University
Information Technologies
Academic Technologies
2129 North Campus Drive
Evanston, Illinois 60208-2850
(847) 491-4055

bef@nwu.edu
http://charlotte.acns.nwu.edu/bef/

DCE and DFS came to Northwestern University in 1993, when the decision was made to replace our monolithic IBM 4381 running VM/CMS with a more modern computer architecture. The IBM 4381 was used almost exclusively for Social Science Research. Email services had already moved off to a variety of other kinds of systems, and the Physical Sciences and Engineering were using a wide variety of computing systems, in a very decentralized manner. It was time for the Social Sciences to catch up.

IT management wanted to build on existing UNIX system management expertise, and also to exploit the $3M investment just completed that extended the campus network to every building, to most faculty offices and to each student's pillow. It was very important to distribute the costs of the new architecture in a decentralized manner. We did not want to have to raise the political capital for million-dollar mainframe-style purchases again. Instead, we wanted an architecture where individual faculty, research groups and departmental entities could make hardware and software purchasing decisions on their own, and be in a position to integrate their acquisitions into the whole.

We were attracted to open systems architectures which exploited the network. I designed an infrastructure that combined the strengths of a central IT organization with the strengths of the decentralized faculty and research groups. IT would provide central file services, tape services, operations, networking services and system management expertise. These would be combined with a number of workstations already in offices spread around the campus. We had to support a combination of UNIX workstations, Intel-based PCs running Windows, and Macs. The personal computers were initially viewed as terminal emulators, but we expected to integrate them into the computing architecture more and me as software and hardware improved.

The participation of the faculty in the planning process was very important. A large focus group of about 20 significant users met twice, to gather information about their needs and to listen to their anxiety about technological change. A smaller group was then formed to focus on specific areas. Functions were identified that were most important to the faculty for their research computing needs.

The new architecture had to serve six functions:

• Processing Large Files

Social scientists tended to work with files that are limited in size by what will fit on one magnetic tape. Those who could manage to work with multi-volume tapes generally stopped after two or three volumes. So, with the IBM mainframe, large files generally ranged in size from 200MB to 500MB because an IBM 3480 cartridge held about 180MB of data. Nine-track tapes were in the same ballpark.

Researchers also had large *collections* of files, some ranging to one or two GB. The exact totals were hard to estimate because so many of their files were in poorly documented tape collections.

The mainframe environment was too constraining. We didn't have enough disk space available, and we didn't want to invest in more (obsolete) disk technology.

• Access to Tapes Created by IBM CMS and IBM MVS Systems

Most datasets arrived from the outside, and most of the sources of data for social science research were still using IBM mainframe systems. It was clear that we would have to continue to support mainframe tapes, even if we moved off the mainframe.

• Maintain a Data Archive

We had over 1,500 tapes in a social science and economic and business data archive that were essential to research and graduate teaching in the social sciences. New data arrived on tape. The national archives were not yet able to exploit the Internet to deliver their services.

• Provide Easy Access to a Wide Range of Software and Computing Services

The mainframe was a single source for a number of statistical computing packages, as well as compilers and programming languages. We wanted to *expand* on the number of available software packages, as well as expand our ability to meet the demand for CPU cycles and disk I/O. And we wanted to integrate high-end personal computers, too.

• Move Large Amounts of Data To and From Outside Researchers

We needed higher bandwidth and more convenient ways to exchange data with the outside world. The networking facilities on the CMS system were viewed as archaic, even though they were equivalent to those on UNIX systems. The real problem was the need for IBM 3270 terminal emulation to use the mainframe. A major culture clash arose around this issue and fueled the emotions that forced the mainframe out.

• Support Large Classes

In our context, a large class has at most 120 students, and they all are assigned to do the same task at the same time. So we had to be able to handle surges of use with a frequency of two to three times a week. And there was an end-of-quarter surge as well, as final papers came due. Classes like these placed heavy demands not only on CPU and I/O capacities, but also disk storage (students make lots of copies of the same file!), printing services and computer lab access.

The Role of DCE and DFS

It was clear to us that we needed a wide area file system that was effecient and secure. NFS was discarded up front because it was perceived as too inefficient to span the campus delivering files larger than 200MB on a regular basis. AFS had a good reputation among the national supercomputer users. It was an established wide area file system, and it used Kerberos for its security. In an effort to further check it out, I attended the AFS User's Group meeting in the fall of 1993, and right up front it was announced that AFS was nearing the end of its product life (they gave it 3 years, as I recall) and that DFS was going to succeed it!

DCE and DFS were described as better-integrated into the operating systems than AFS because AFS had no code in the kernel, and DCE and DFS did. Where AFS was provided by one vendor (Transarc), DCE and DFS were provided by many UNIX vendors, and PC products were also in the works. Interoperability between vendors was important, and was proved on an annual basis by the Open Software Foundation in the form of an Interoperability Festival.

Perhaps most important (looking back) was the thought of implementing AFS only to have to convert to DFS within three years. Our faculty were already heavily stressed by the prospects of converting from CMS to UNIX over the course of a year. Another conversion, however minor, would not have been accepted by the faculty if it were built into the implementation plan up front.

The Importance of a Test Cell

We decided to investigate DCE and DFS, and determine if the products were mature enough to put into production. We grabbed two HP 715/33s and set up a two-machine DCE cell running DFS. We found lots of problems, but HP was very supportive, and our work progressed well enough to move ahead. But we always had the notion that, should DCE or DFS fail us, we could fall back to AFS or even NFS.

It's very important to set up a "sandbox" DCE cell when you are starting out. You need the opportunity to experiment, to try things and to back them off, or even to start over! There's much to learn, and you won't learn efficiently if you're always hesitating to try things out, in fear of loss of production time. Later on, as new versions of DCE and DFS have been released, we've gone back to our sandbox cell to try them out, develop migration scripts and prove our methodologies.

The Initial Configuration

Obviously, we chose DCE and DFS as the underlying technologies to replace the mainframe system. Those products offered us high security in a networked environment. They were scalable in the sense that we could transparently add server systems to the DCE cell to meet increasing demand for performance, storage capacity or CPU capacity. And the ability to scale offered us the opportunity to extend those services campus-wide in the future. At the very least, they offered the opportunity to learn how to manage a high-performance distributed computing environment on a relatively small scale. No matter what turns the technology would take in the future, the knowledge gained in managing such an environment would be valuable.

Our initial configuration was viewed as a starting point from which to add on. It consisted of three computing systems designed to satisfy the important functions enumerated above, all connected to a high-capacity network.

• An FDDI Ring for High-Capacity Network Bandwidth

We were fortunate enough to find the funds to purchase an FDDI concentrator and FDDI boards for our high-capacity servers. This FDDI ring was in turn attached to the campus FDDI backbone via the FDDI concentrator. So we had a big pipe directly attached to the other big pipes on campus.

• DCE and DFS Server

AN HP 9000/H50 (99MHz) with 20GB of disk storage was purchased to be the DCE security and cell directory servers, as well as the DFS server. It has an FDDI connection to the network, and all of the disks were Fast-Wide SCSI 2 disks. The machine was configured to handle substantially more disk storage in the future. This system stands alone. Only administrators can login to the machine.

• Magnetic Tape Library Server

An IBM RS6000 model 34H was purchased to run our tape library software (REELlibrarian from SCH), and was equipped with an 8mm drive, a single 9-track drive, and a channel connector to an IBM controller for a bank of IBM 3480 tape drives. The channel connector cost around $4,000 and gave us access to up to four 3480s, each of which would cost at least $20,000 (at the time). Our initial budget included one 3480 drive. The savings enabled us to purchase the FDDI concentrator.

We use DCE to manage **/etc/passwd** on the tape server. The tape library system requires an **/etc/passwd** file that is consistent with the same file on all of its client systems. We use the DCE **passwd_export** utility to manage this task. An **/etc/passwd** file is created once each day in such a fashion that normal users are defined, but unable to login to the tape server machine itself. Instead, users access the tapes thru client/server software on other DCE systems.

• A General-Purpose Compute Server

We started with just one machine for our users to login to, an HP 735 (99MHz) running as a DCE and DFS client system. Home directories were put in DFS, and a DCE login was performed as part of the login process. The system was configured with a 500MB DFS cache, and a 3.5GB scratch disk.

With home directories in DFS, we planned to add compute servers as required. Indeed, not long after we went into production, a second compute server (an HP J200) was purchased by a research institute. That machine was restricted to about 30 users who were members of the research institute. Because the statistical software was installed in DFS and licensed for general use, users of the second compute server were able to access the same suite of software as users of the first server.

The Disk Cache Architecture of the Compute Servers

Two disk caches are important. The DFS cache holds files resident in DFS. Only chunks of the files actually in use are transmitted over the network and stored in the DFS cache. When those chunks are accessed again, they are found in the local cache and no network retransmission is necessary. If a DFS file is changed, only those bytes that are changed are sent back to the DFS server. A technique called *token management* is used to coordinate file changes among DFS clients. If a byte is modified in a DFS file, the token corresponding to that byte is revoked from every DFS client system using that file. Those clients can only access that byte after the new value is transmitted from the DFS server.

A second cache was designed to enhance the performance of statistical packages such as SPSS and SAS. Those programs tend to copy and merge data sources into temporary files called scratch files or work files, and then they pound on those temporary files, using them over and over again for the duration of the job. We would take a big hit in performance if we were to create those temporary files on a DFS volume because of the network transfers involved.

Instead, each compute server is configured with a local high-speed disk called a *scratch disk.* The statistical packages are configured to write their temporary files on that scratch disk, removing them when they terminate. We keep this scratch disk separate from the traditional **/tmp** and **/var/tmp**. It is named **/scr01** on all systems, and each user is given a subdirectory corresponding to their login name. Huge files are sometimes created in this scratch disk.

We try to configure our DFS caches to hold as much storage as possible. For our general-purpose systems with 600 accounts, a 1GB cache is the current norm. Some single-user systems have a 100MB cache. And some systems which have only minimal use of DFS have 50MB caches.

At times, the scratch disks are limiting factors in the use of these systems because they fill up. As such, we made them as large as possible. Under HP-UX 9.x we used 4GB scratch disks. When we upgraded to HP-UX 10, we increased the scratch disks to 8GB whenever possible. Hourly cron jobs keep the scratch disks clear of occasional orphaned files left behind when processes abort, or are killed by users.

Exploiting Compressed Files

In addition to the hardware and software architectures designed to deal with large files, I also work intensively with end-users, teaching them to use **gzip** and **gzcat** to pipe decompressed data directly into the statistical packages. As a rule of thumb, the average raw data file for social science research, consisting mostly of strings of ASCII numbers, will compress about 80% using **gzip**. SAS datasets, which are proprietary binary files, compress even better. Many of them compress 90 to 95% in size!

In the case of our large data library, all of the disk-resident files are stored in compressed form. Users are not allowed to access library tapes. Instead, the files are transferred to disk for them. Once transferred, the files remain on disk. Currently, our archive collection is about 3.5GB in size. Assuming an average compression rate of 80%, that works out to 17.5GB of uncompressed storage.

Managing Statistical Applications Using DFS

All statistical programs are installed in DFS directories and added to the user's PATH at login. There is no need to maintain copies or depots of the software on individual client systems. Given the efficient DFS caches, once a program is transferred into the cache, execution is as efficient as if it were installed on a local disk.

Software licensing practices vary widely among the various vendors. We make an effort to obtain licenses that provide the widest access possible. Still, some packages are licensed on a single-CPU basis, and others are licensed on a per-user basis. DFS works quite well in each case. For the single-CPU licenses, I still install the package in DFS. Then I run license managers as required, either on individual workstations or in a cell-wide fashion. A similar situation holds for software licensed on a per-user basis.

Still, I have to support people who want to install their own software locally. I do this by constructing the user's PATH from three sources, in the following order:

1. **/etc/PATH** is used exclusively for HP system software
2. **/etc/PATH.LOCAL** is used for locally-installed software
3. **/.../nwu.edu/fs/opt/@sys/PATH.DFS.BIN** for software installed in DFS

By using this order, system software overrides anything installed locally, and locally installed software overrides anything installed in DFS. As such, a professor can install his own Beta copy of a package without worrying that it is the same package as one made available campus-wide thru DFS. Of course, the login scripts can easily be modified to change the order of precedence.

The same methodology is used to manage **MANPATH**.

As a system manager, I can install and test a new version of a package in DFS, and when it's time to release it, all I have to do is modify the third path file above. The next time a user logs in, that user will be using the new version of the package.

The Role of DCE Integrated Login

DCE credentials are required to access files in DFS. You obtain your credentials, or DCE identity, by performing a **dce_login**. Before DCE Integrated Login became available, we had to **dce_login** as an additional step after UNIX **login**. That made it somewhat inconvenient to work with home directories in DFS, because the DFS files were inaccessible until DCE credentials were obtained. (The workaround is to grant world read access to the dot-files required for login, enabling the shell to be properly started.)

DCE Integrated Login combines UNIX login with DCE login as one apparent step in the HP-UX login process. It uses PAM, the Pluggable Authentication Module, as the security framework, and is quite flexible as a result. We use three different login configurations at NU.

• DCE as the primary login technology, with no fallback. Our largest compute servers rely on the DCE security service for login. If the security service is unavailable, users won't be able to login. Given that home directories reside in DFS, that's just fine. And in the case of the **root** and **operator** accounts, login is overridden by a shadow password file. Those two accounts can login even without the DCE security service.

• DCE as the primary login technology, with UNIX fallback. Sometimes, faculty members want to establish accounts that do not use DFS home directories. But they also have accounts that *do* use DFS home directories. In this case, the DCE login is performed first, and if it fails, a UNIX login is performed (which handles the accounts not resident in DFS). Again, local **root** login is controlled from a shadow password file.

• UNIX as the solitary login technology. The Statistics Department systems are somewhat self-sufficient, and only use DCE and DFS to access the large suite of software provided. DCE credentials are not needed to run the software, so no DCE login is required. If we make our DCE cell globally available some day, we'll have to tighten up access rights to the software suite.

It should be noted that **local root** access to a DCE or DFS client system gives you no special access rights to files in DFS. The local root user cannot access any of the home directories, for example. The DCE principal named **root**, on the other hand, *can access every file resident in DFS.* Guard your **DCE root** password carefully.

It is also important that the DCE Security Service be available at all times. We run our Security Service on two different machines, as replicated servers. If one becomes unavailable, the other fills in transparently. As the cell grows in size, we can continue to replicate our Security Servers to meet increased demand.

Using Groups to Control Login to a Compute Server

We use DCE groups to control login access to compute servers that are configured with DCE Integrated Login. For machine *hostname*, all members of the group *hostname* are allowed to login.This is accomplished by using group membership to construct the **/etc/passwd** file on the local machine. If your account isn't in the **passwd** file, you can't login. The file is updated hourly using a **dcecp** script named **account_export**. (**dcecp** is a DCE-aware version of **Tcl**.) See my web page for a detailed explanation.

DCE Security Service and Cell Directory Service

For any production cell, you should separate the DFS services to their own machines, and put the security server and CDS servers elsewhere. Currently, we have one machine, an HP H50, running our

DFS services. We put our security and CDS servers on two other machines, each machine running both servers. Our DCE security and CDS servers are said to be *replicated.*

Replication offers ease of maintenance as well as improved performance and reliability. I recommend that you start a production DCE cell with at least two machines to run your CDS and security services. Plan to add a third machine later on if you can't afford it up front. For cells of a few thousand users, I would think that these services are not CPU-intensive, but they are sensitive to memory and swap space, so pay close attention to your hardware configuration and leave room to grow. And in the case of CDS, the default **dcecp user** object creates a CDS directory for each user. Change the object so it does not create the CDS directory. You won't want ordinary users putting things into the CDS.

The Cell Configuration Today

Our DCE cell for social science research has grown dramatically over the past year. We have about 600 user accounts. Our one DFS server now has 66GB of disk storage, and we have two machines each running DCE security and CDS servers. We have added a second general-access compute server (a two-CPU J210), and we have added ten DCE client systems scattered around campus. Four more are in line to be added this summer. We'll have a total of 17 compute servers by September. A separate machine has been acquired to hold the data archives for the library. And, we have purchased an IBM system to serve as a second DFS server. We'll put that system into a test cell first, so we'll know what we're doing when we put it into production. Currently, DFS files are backed up to a DLT drive as regular UNIX files using **fbackup**.

The Role of Windows NT

Windows systems are very prevalent in the social sciences at Northwestern. Within Economics especially, Windows systems are used in a manner equivalent to our UNIX systems. There is a very large overlap of statistical software between the two systems. We plan to exploit NT 4.0 versions of DFS clients as they become available this year. This would enable NT users to access their files in DFS just like the UNIX users do. And further down the line, we might be installing NT versions of the statistical software in DFS in a manner very similar to what we're doing today for HP-UX users.

Issues and Concerns

HP's DFS product is rather old technology, and is being replaced by their Enterprise File System (EFS). I view EFS as a much more complete and up-to-date implementation of DFS on HP-UX. As such, I am looking forward to installing EFS, and pairing it off against IBM Transarc's DFS. Once we upgrade to EFS and IBM Transarc's DFS, we will have the opportunity to employ backup filesets for additional data integrity, as well as use DFS Backup servers to back up *filesets* to tape. And, we'll be able to replicate the filesets containing our statistical software, thereby improving reliability and availability.

It is a must that you keep up with maintenance if you are using HP's DCE and DFS products. This means running HP-UX 10.20, and it means that you must keep up with current system patches as well as DCE patches. And the additional security brought by DCE is worth very little if you don't keep up with HP-UX UNIX security patches, CERT advisories and so forth.

HP's Support Services are very important to DCE and DFS customers, and they have sometimes let us down. When placing a support call, it is best to insist on talking to a DCE analyst without specifying anything else in detail. Once you reach the proper analyst (don't let them give you a CDE analyst!), you can then move on to DFS questions (which usually require another referral).

HP is slow to respond to DFS calls especially, and I have the sense that their support analysts lack not only manpower, but also hardware resources. Too often, I've heard that the HP analyst is not running current levels of DCE or DFS, and can't duplicate the problem I'm reporting.

HP can best improve their support services by using DCE and DFS more internally. They need to learn to live with their own products, and spread that knowledge around within the corporation. Given the scalability of both DCE and DFS, a good candidate would be HP's web servers. They should store their web pages in DFS, replicate their DFS servers, and run their web clients as DFS clients with large caches. The DFS replicas can be spread all around the world to distribute network bandwidth demand and eliminate bottlenecks.

The Importance of Change Management

I can't overemphasize the importance of managing change in our environment. Our success is largely grounded in the up-front time spent with the end users who had substantial investments in their computing infrastructure. Their participation in the planning and implementation processes helped carry those investments forward, and it also gave greater integrity to the project in the eyes of upper management. The net result was continued support and funding from the administration, as well as the faculty as individuals and as research groups.

On the other hand, we have had a certain lack of success because we did not spend enough time helping the mainframe system managers deal with their change issues. They reside in a separate IT organization, in another building, and weekly and later bi-weekly meetings did not suffice. It takes very large measures of patience, care and understanding to deal successfully with issues of change.

Summary

Northwestern University's DCE cell for Social Science Research is very real, and very successful in many dimensions. It is not without its problems, especially regarding DFS reliability. It is a work in progress. It is a stepping stone into the future.