# CRI Distributed File Systems Update
# NFS, NFS V3, DCE/DFS and SFS

*Brian Gaffey*, Cray Research, Inc., 655-F Lone Oak Drive, Eagan, Minnesota 55121

**ABSTRACT:** *This paper provides an update on CRI's distributed file system products. The paper covers the current features, planned features and future plans for all of CRI's distributed file systems. It covers our UNICOS products and plans as well as our plans for UNICOS/mK Additionall, each of CRI's product offerings is compared and contrasted. The paper covers how each of the product offerings brings an unique advantage to customer problems. Topics covered include transparent access, performance, reliability, scaling, security, data integrity and standards.*

## 1 Introduction

This paper compares the different distributed file system technologies offered by CRI. An overview of the technologies is presented by first showing how each contributed to CRI's SuperCluster product. The CRI distributed computing framework is presented in order to contrast how the products are implemented. UNICOS and UNICOS/mK requirements are also covered. Finally, each of the products is compared in terms of general requirements. The general requirements we've used are:

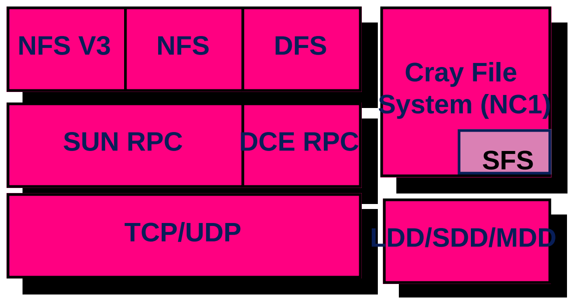
- transparent access
- performance
- reliability
- scaling
- ease of administration
- data integrity
- security
- standards

The products being compared are the following:

- Network File System (NFS): This widely used mechanism for sharing data between nodes provides an open solution.
- OSF Distributed File System (DFS): This emerging standard for sharing data between nodes provides an open solution that offers increased performance, transparency and security over NFS. DFS is based on the OSF Distributed Computing Environment (DCE).
- Shared File System (SFS): This high performance, HIPPI-based common file system is shared among CRI UNICOS systems only. HIPPI Network Disk Arrays are used as the storage media. SFS offers resiliency - if one system fails, the other systems can still access the data - and the highest level of performance for suitable applications.

## 2 Distributed Computing Framework

NFS, NFS V3, DFS and SFS all co-exist within the distributed computing framework. A system can have all or a subset of them executing at the same time. However, there are differences in their implementations and features.



NFS and NFS V3 share the same RPC mechanism. Both can run over the ONC RPC or they can be enhanced to support a Kerberos version 4 style of security called auth_kerb when ONC+ is installed. ONC+ also supports NIS+.

DFS uses a completely different RPC mechanism from NFS. DFS's RPC is fully integrated with the DCE security service which is based on Kerberos V5. The DFS RPC mechanism also runs over UDP/IP but it has been enhanced to include retransmission and flow control. In other words, it behaves like a connection oriented transport layer which allows for greater performance over WANs.

SFS is not a networking protocol. It does not run over a RPC of any kind. SFS communicates with its peers via a shared media - usually a disk. Its mechanism for communication are the systems disk drivers.

## 3 NFS and NFS Version 3

NFS is the most widely used network-based filesystem.Open Network Computing (ONC) is a suite of products from SunSoft. This suite contains NFS, NIS and other distributed applications. ONC+ is an enhanced version of ONC. It contains: NIS+, NFS version 3 (V3), AUTH_KERB RPC authentication and a new mount protocol.

NFS V3 is available in UNICOS 9.0. It contains support for 64 bit file descriptors, larger packets, new access permission mapping and reduced network packets. Our experiments have shown NFS V3 reduces the amount of CPU time spent in the server.

## 4 SFS

UNICOS SFS allows the UNICOS native file system (the NC1 filesystem) to be shared among multiple UNICOS systems. All of the features present in the NC1 filesystem (such as device striping, mirrored file systems, asynchronous I/O, etc.) are supported through SFS. Under optimal conditions, and for suitable applications, SFS can deliver near NC1 performance levels (95%). It therefore potentially offers the highest level of data-access performance.

SFS requires media that can be shared by each UNICOS system and an arbitration device that controls access (via semaphores) to the media from the individual nodes. Today, SFS supports HIPPI-based Network Disk arrays (CRI ND40 devices) as the shared media and uses a arbitration device built into the ND40 controller. All participating UNICOS systems and the network disk array are connected on a switched HIPPI network.

## 5 DCE/DFS

DFS is a part of the Distributed Computing Environment. CRI's latest release is the 1.0.3 version of DFS. That release supports a number of performance enhancements to the base DFS and an integrated login feature for UNICOS. DFS 1.1 which will release later this year, will provide support for hierarchical cells, delegation, multi-level security (MLS) and Kerberos Version 5 interoperability.

The primary mechanism that DFS uses to obtain high performance is caching. DFS uses local disk based caching. When chunks of files are read from the server over the network, they are stored in files on the local system. Subsequent accesses to that data use the local cache rather than pulling it over from the server again. DFS's disk based caching approach provides performance that is comparable to local file system performance, which is much faster than actually transferring the data over the network from the server system each time a file is accessed.

Read access to the DFS cache comes in two flavors, *hot cache*, which refers to the situation where data is already present on the client and *cold cache* where it must be fetched from the server.

DFS is divided into a client and server. The client is commonly referred to as the Cache Manager (CM). A DFS machine can act as either a client or a server or both. As well as having the DFS kernel extensions, a DFS machine must run DCE core services (as described above). Along with the cache, one of the primary advantages to DFS is the global namespace. In other network file systems, the path to access a file on one machine may be different than the path on a different machine. This can make it difficult for applications to locate files. In DFS however, the path to a given file is the same for *all* DFS clients.

## 6 SFS with DFS

We've combined DFS and SFS to provide a smooth performance profile for cluster users. A smooth performance profile includes good performance over shared media regardless of the blocksize. The goal is to combine DFS's performance for small blocks and SFS's performance for large blocks. Another advantage of this approach is that DFS provides an "open systems" access for non-CRI platforms.

SFS also facilitates the construction of fault-tolerant cluster data configurations. A 'heartbeat' mechanism held in the semaphores allows healthy cluster nodes to detect outages of other cluster nodes. Recovery can then be automatically initiated so as to allow uninterrupted data access by the remaining cluster nodes

## 7 SuperCluster

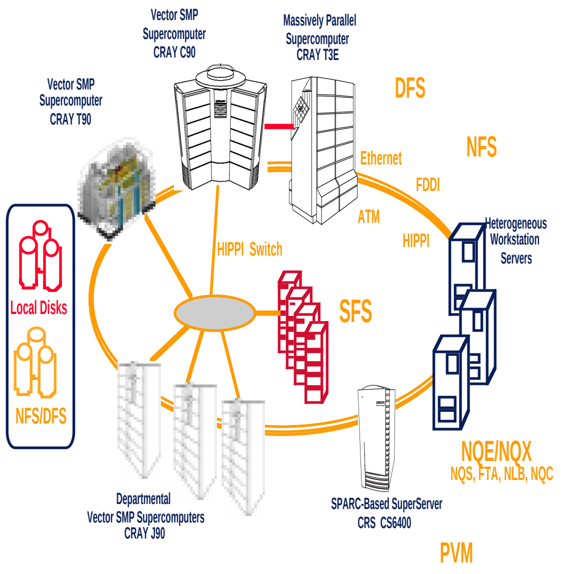
At the center of the cluster concept is the ability to control and balance the execution of all types of work (batch, interactive, and parallel) within the cluster. We have chosen CRI's Network Queuing Environment (NQE) to perform this function. NQE runs on all CRI systems and on most popular UNIX workstation systems and servers.

Given that work may execute in any part of the cluster, a key requirement is high performance, transparent access to data from any node. NFS, NFS V3, DFS and SFS all play a part in meeting this requirement.

## 8 Requirements

Since CRI supports NFS, NFS V3, DFS and SFS we are often asked which is the best. There is no easy answer to that question. Each of the products has strengths and weaknesses. The following list is an attempt to compare the products given a high level set of requirements.

**Transparent access** is the capability to access data that is not local to your system via names that appear to be local files. This is the minimum requirement needed for a product to be considered a distributed file system. All of CRI's file systems


Vector SMP Supercomputer CRAY C90
Massively Parallel Supercomputer CRAY T3E
Vector SMP Supercomputer CRAY T90
DFS
NFS
Ethernet
FDDI
ATM
HIPPI
HIPPI Switch
Heterogeneous Workstation Servers
Local Disks
NFS/DFS
SFS
NQE/NQX
NQS, FTA, NLB, NQC
Departmental Vector SMP Supercomputers CRAY J90
SPARC-Based SuperServer CRS CS6400
PVM

satisfy this requirement. However, DFS makes use of a non-standard name format "/:/". This can be hidden with links.

**Performance** is the ability to provide data to a consumer or to move data from the producer. The goal of any distributed file system is to provide performance equal to the local file system. This category is one of the most difficult to quantify since the number of options is so large. For example, NFS and NFS V3 provide performance that is adequate for LANs. However, their performance in WANs and SANs is usually not very good. Their performance in WAN/SAN environments can be made slightly better by using TCP instead of UDP. By using TCP, the worst case scenarios are improved not the overall performance.

DFS can achieve performance equal to the local file system via use of its caching techniques. When an application using DFS is accessing data in the DFS clients cache then the cache is called hot. The trick is to keep the cache full of data for consumers and to let producers of data continue while the dirty data is flushed to the server. CRI has spent a lot of energy improving DFS's performance and we think it is the fastest in the industry.

SFS can achieve performance equal to the local file system. This is especially true for well formed IOs, locked accesses and read-only file systems. In the latter two cases, SFS bypasses its locks on disk and uses the local file system directly. Well formed IOs are IO which are larger than 500K bytes and on sector boundaries. SFS also does well for applications which don't perform many meta-data operations (e.g. create files, change directory structures, etc.). However, SFS performs very poorly in all other cases. For the right type of application SFS is a good performer but it is not a general purpose solution.

**Reliability** consists of architectural extensions or assumptions built into the products to enhance availability.

NFS and NFS V3 are based around the concept of a stateless server. This has been very successful in helping NFS deployment. Partly, for this reason and to keep the product very simple, no additional features have been added for availability. So, if a NFS server fails then the files associated with that server become unavailable.

DCE and DFS are also client/server based. DFS is designed to be very stateful. A DFS server maintains the state of all active clients. When a server starts it goes thru a period called token state recovery. During this period all clients are contacted and asked to tell the server their state. To improve availability, CRI has implemented DFS server failover. When a failure occurs a command is executed which causes the CDS and FLDB databases to switch access to the new server. The new server has access to the same files as the original server thru the use of SFS or DFS replication in the future.

SFS, as stated earlier, is not a networking product. It is not client/server software. In a SFS environment, all participating machines are peers. Therefore, no single SFS node is a point of failure. If one SFS node fails the others can continue without intervention.

**Scaling** is the ability to have a large number of clients per server and a large number of servers per enterprise. NFS due to its simple design puts a lot of burden on servers. The number of clients per server is therefore limited. DFS supports a much higher ratio of clients to servers. DFS uses caching to decrease the load on the server and replication to improve scaling. SFS, since it is not a networking product, is designed for a small number (currently 64) of nodes.

**Ease of administration** is usually expressed as the number of systems an administrator can manage and the amount of training the administrator requires. In this category, NFS is the simplest to learn. NFS tends to be "client-centric" in that configuration changes often require administrative work on the client systems. DFS is more complex than NFS. For DFS an administrator needs to understand the concepts behind DCE and DFS. Specifically, the use of DCE's directory service (CDS), security service and DFS style naming. The learning curve is high but once mastered, most administrative tasks can be done using the databases. And DFS clients do not require updating. SFS is relatively simple but it does require an understanding of file systems and network disk arrays.

**Data integrity** is a measure of how well the file system protects user data. NFS's Lockmanager provides advisory locking. Using the Lockmanager, applications can protect access to shared data. DFS uses a technology called token management. With token management, all client accesses to data are protected with the tokens. Token management is transparent to the user application. SFS uses a concept similar to tokens called semaphores. Semaphores are an on-disk shared lock which SFS manages transparently for users.

**Security** refers to the level of authentication and authorization performed by the file system or on the file system's behalf. NFS enforces a UNIX style or level of security. This is built around user ids (uids) and group ids (gids). CRI has extended this to include NFS ID-Mapping. This imposes an additional test on NFS access. ONC+ includes support for auth_kerb RPCs. This incorporates support for Kerberos version 4.

DFS provides integrated security via the DCE security service. DCE's Security services is built into DFS. The DCE security service is based on Kerberos version 5. Both NFS and DFS enforce CRI access control lists if they exist on a file. Additionally, NFS was part of the B1 evaluated system for multi-level security. DCE will support MLS in the 1.1 release.

Access to SFS file systems is identical to access to a local file system. SFS has not been tested in a MLS environment.

**Standards** are industry agreements. NFS is a defacto standard for distributed file access. DFS is part of OSF's Distributed Computing Environment which is an emerging standard. SFS is CRI proprietary.

## 9 UNICOS requirements

UNICOS requires that the file systems support be integrated with other parts of the system. Specifically, the CRI extensions to UNIX. NQE is the extension to UNIX that allows batch work. NFS, DFS and SFS have all been integrated with NQS. NQS can read and write to any of the file systems. Also, there are additional features for later releases of NQE that apply to cluster security. Please see the paper on DCE and UNICOS Security in these proceedings.

Quotas are used in UNICOS to control shared access to limited resources such disk. NFS and DFS enforce UNICOS quotas but can not display them to non-CRI systems. DFS does not support LFS style quotas. SFS does not support quotas of any kind.

Checkpoint and restart of jobs with active file system activity is supported in UNICOS 9.0 for all file systems.

## 10 Gigaring

Gigaring support is transparent to file systems. No additional changes are required to file system code. However, Gigaring affords us the chance to optimize our products. We will examine support for fiber channel connected disks as a shared media. We hope this will improve the latency of SFS IOs and allow for better performing small block IO.

We will also experiment with NFS and DFS access to shared media between systems. This architecture may allow for true third party transfers.

## 11 UNICOS/mK Requirements

UNICOS/mK requires the file system components to be serverized or split from the rest of the operating system. In this architecture, all of the file system components reside in the Object Manager. Given a MPP architecture this may result in a bottleneck since we do not have multi-threading. To address scaling of IO we used multiple PEs of a MPP instead of multiple CPUs of an SMP.

The File System assistant provides most of the file system processing (after open) on the user's PE. DISTIO allows a single PE to request IO on behalf of other PEs. Remote mount allows file systems to be split at their mount point over multiple PEs. NFS and DFS will use remote mount to provide a dedicated PE for processing.

## 12 Summary

CRI supports the major distributed file systems available in the industry. We also provide support for a proprietary high performance shared file system. Each of the file systems has advantages and disadvantages. Taken as a whole they provide a well balanced set of alternatives for data access.