Shared File System Experiences

Paul Anderson, DOD and *Tony Picco*, CRI

ABSTRACT: *One year ago, we presented a paper on CRI's Shared File System product and how our site had learned to use SFS to take advantage of its strong points. A year later, we have numberous SFS experiences to share -- some of them better than others. A number of hardware and software issues have been resolved, but reliability continues to be a problem. This paper will address the problems we have seen over the year and how to try to avoid them. We will also evaluate SFS as a product in its current implementation.*

Introduction

At the Fall '95 CUG, my collegue, Denise Underwood-Hannagan, presented our experiences with CRI's Shared Filesystem (SFS). As she indicated, we got off to a rough start, but things have progressed since then and one of the main topics of this talk is to provide an update on the status of SFS, one year later.

To review the basics of SFS, this is CRI's attempt at providing the capability of accessing filesystems across multiple CRI platforms. The primary advantage provided being reduced data duplication and so a savings in disk space and backup requirements. Additionally, as was presented last year, SFS's performance gain was with large block I/O primarily with files which were written once and read often. With an understanding of the basics in mind, we set out to put SFS into a production state.

Hardware Configuration

Our original configuration consisted of two Model-E machines, one H-SMP (a semaphore device), an ND14, and an existing HiPPI network. The first machine, the semaphore box, and the ND14 were connected to the same NSC PS32 HiPPI switch. The second system used an additional PS32 and fiber optics in order to get to the common HiPPI switch.

The system configuration has now changed to where there are multiple (more than the original 2) Cray systems accessing SFS, which now resides on an ND-40 disk (see Figure 1). Some of the CRAYs are connected to the same NSC PS32 switch as the ND-40, whereas the others use different PS32s and fiber optics to get to the ND-40's switch.

The ND-40 has an internal semaphore device and is connected to an UPS (uninterrupted power source) system. The ND-40 is configured with a RAID-1 section for the Shared Lock Region (SLR) and the meta-data, with the filesystem itself being a RAID-5 section. The reasons for going to ND-40s was that they were more reliable, faster, had an internal semaphore device, and had a larger capacity.

Another addition to the configuration was a second ND-40 containing another shared filesystem. As seen in Figure 1, this tested the capability of one system accessing two different SFS facilities.

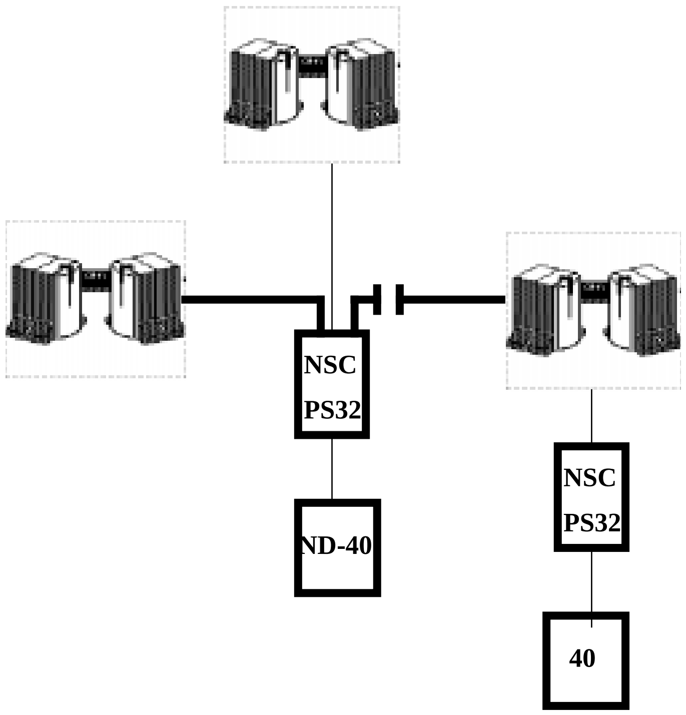

Figure 1

Of course, this change in configuration was done in a logical and sequential method to help minimize risks and identify problem areas.

The first step was to replace the old ND-14s with the newer ND-40s. Once the ND-40s were in place and the configuration successfully re-tested, we began attaching additional systems. Since the systems being connected reside in different areas, we soon ran into a transfer rate performance problem. This problem was tracked down to a bandwidth capacity limitation between the two areas. This bandwidth limitation was not a problem with SFS, but the addition of SFS usage caused the network to reach the capacity limit. To correct this, we connected additional fiber between the two areas.

Unfortunately, once we got past this problem, a logic flaw in the hardware design of the PS32 switches was identified and needed to be corrected. Again, not a SFS problem, but one which hampered SFS production capability.

Now with the switches fixed and the increased number of fiber runs between areas, we are able to take advantage of isomorphic addressing. Isomorphic addressing allows the switch to better balance the network load. Previously we were using source addressing which is point-to-point routing. Isomorphic addressing provides us with better alternative routing capabilities and makes better use of site's networking resources.

Software Aspects

Once hardware connectivity was estabished, we focused on software aspects. UNICOS 9.0 became available in the midst of this reconfiguration attempt. Upgrading to 9.0 was advantageous since 9.0 incorporated SFS mods that we had applied separately to the 8.x release. Unfortunately, the 9.0 implementation of SFS was not compatible with the UNICOS 8 release we were currently running. Therefore, each SFS associated system needed to be upgraded for SFS to continue to function. In addition, because our configuration of SFS uses the HiPPI interface, we couldn't minimize our risks by using the UNICOS under UNICOS feature. This upgrade procedure took time, coordination, and, most of all, patience on the part of the system administrators, users, and the system support group.

Though the 9.0 version of SFS did prove to be a better and more robust product than its predecessor, we still ran into several problems. These problems included HiPPI network errors, ND-40 drive errors, shutdown and restart script problems, and the need for a standard set of operating procedures. Most of these problems have been resolved through the efforts of our local site analysts and Eagan developers. However, it has been a long and painful process. In some cases, correcting one problem uncovered another.

HiPPI network errors occurred due to hardware and software problems. Fault tolerance evaluation was done to determine the extent that the SFS software would be able to correct itself from errors, such as timeouts, occurring on the network. We performed controlled testing and determined that the configuration could handle an outage of up to 30 seconds. This amount of time was considered acceptable, since at our site any outage greater than 30 seconds would imply more serious problems. Additionally, better handling of errors, resulting from HiPPI network problems, has been done. When a HiPPI error occurs, recovery is triggered in the software. These recovery cases are exercised and in some cases needed additional code. Lengthy evaluation has been done on-site and corrective code written to enhance the recoverability in these cases. One example was that the kernel would flood the IOS with retry packets when a HiPPI error occurred. A local modification was made to limit the number of retry packets. Another modification was made to clear a semaphore hung due to a HiPPI interrupt.

A combination of hardware and software problems also caused ND-40 errors. We simulated power fluctuation and loss to the ND-40. Initial tests highlighted a problem that the HiPPI ports on the drives were not being reset correctly after a power bump. We decided that to reduce the risk of power flucuations to the ND-40s, we would connect them to an available external UPS. This connection has substantially reduced the errors encountered from the ND-40 device.

From a software standpoint, a timeout difference between the IOS and the HDDTSLEEP interrupt routine caused the I/O to the ND-40 to be interrupted. The straightforward change of the timeout setting to be the same in each corrected this problem.

On the whole, the release of the 9.0 version of the SFS software greatly improved the normal operation of the facility. However, some adjustments to the startup and shutdown scripts had to be done. In "sfs_stop", SFS was not being unmounted correctly. This was corrected and tested at site. With the startup script, the problem was in the start sequence for SFS and NQS (Network Queuing System). With a checkpointed NQS job using SFS, SFS had to be up before the job could be started.

With some of these modifications, installing the fix and testing it uncovered other problems. This made the detection, correction, and testing of SFS a long process. However, the validated corrections have become incorporated into our production facility and also submitted to CRAY development for incorporation into their next release.

As with any facility, a set of standard operating procedures (SOP) is necessary. Initially, due to the developmental nature of SFS usage, SOPs for problems encountered were not in place. This lack of guidance led to confusion and false starts in problem resolution which delayed getting SFS operational after a failure. This delay adversely affected processing and reduced confidence in the facility. SOPs are now in place and accessible by appropriate organizations.

Along the same lines, one additional concern was caused by the use of an automated monitoring system at our site. Capturing, identifying, and resolving the various errors which an SFS configuration could cause was a substantial concern for our operations personnel. Of course, no sooner was one error identified and handled then another would be encountered. However, due to the nature of the automated monitoring system and its database acccess, common errors could be logged and identified SOPs made available to the operator. This made the transition over to SFS easier for operations personnel.

The site continually monitors the activity of the SFS facility and tries to develop timely corrective actions to problems as they occur. As mentioned previously, this information is passed back to Eagen. Additionally, a bi-weekly conference call has been initiated among this site, the Eagen SFS development group, and the other sites using SFS. This has proved to be a very useful mechanism for the sharing of information and timely identification, and possible resolution, of SFS complications.

Concerns

The current usage and performance data reflect the increased use of SFS and the immediate savings in disk space. The user is satisfied with the performance and functionality of SFS. With these advantages working for us, we are in a more secure and robust posture to move forward and put an SFS configuration into production mode.

However, as mentioned earlier, there are some drawbacks to SFS usage. Not all filesystems are appropriate for an SFS configuration. Some filesystem analysis needs to be done to identify appropriate filesystems. Also, the underlying network configuration needs to be adequately designed to handle the increase in network load. Furthermore, the type and configuration of the hardware used for SFS greatly determine the reliability of the facility. Appropriate stability of the configuration and error handling is directly proportional to the user-friendliness of SFS.

A more far-reaching concern is that SFS is limited to CRI-specific architectures and so its applicibility for other sites may not be as beneficial as at ours. Currently there is a limited number of other sites at which SFS is used and at each site its configuration is unique. Additionally, SFS is an unbundled package and a bit pricey (even for our site!). Given these factors, is CRI going to continue to support SFS?

As mentioned previously, our site has a heterogeneous computing environment, with the demanding need for high-speed access to shared data. With new hardware and software products becoming available, perhaps a different, more universal approach is warrented. For example, one alternative could be network-connected devices supporting a generic filesystem structure allowing continual access for shared data from different platforms. Distributed computing (DCE/DFS) with its common namespace and current availability on a variety of platforms is a viable option. And NFS is becoming a possible alternative as it becomes available on more platforms and, hopefully, begins to provide comparable transfer rates What is clear, is that a customer can not wait another two years to have a product which may provide this functionality.

What is SGI/CRI's future direction for SFS? With DCE/DFS and NFS 3.0 (ignoring its current performance problem!), where does SFS fit into SGI/CRI's scheme of a distributed filesystem approach? Has SFS's time past? As budgets tighten, resources become more limited, and a greater empahsis is made on distributed computing, it is concivable that a supercomputer-specific approach will no longer have its place in the competive market.

Conclusion

Overall, the system users have been pleased with the performance and flexibility that SFS now provides. We realize it has its limitations and cannot be used to solve all of the challenges in a shared data computing environment. Yet, for applications that read and write large data blocks and require access to common data sets, SFS can provide the capability to reach the data set and allow a more efficient and economical use of disk space.

As SFS becomes more robust, its use in appropriate areas at our site will be extended. However, we are still concerned about SGI/CRI's commitment to expanding capabilities and supporting SFS.