A Comparison on Current Distributed File Systems for Beowulf Clusters

Rafael Bohrer Ávila[1]
Philippe Olivier Alexandre Navaux[2]
Yves Denneulin[3]

Abstract

This paper presents a comparison on current file systems with optimised performance dedicated to cluster computing. It presents the main features of three of such systems — xFS, PVFS and NFSP — and establishes comparisons between them, in relation to standard NFS, in terms of performance, fault tolerance and adequacy to the Beowulf model.

1 Introduction

With the popularisation of Beowulf-class parallel machines (STERLING, 2002) and the constant improvements in technologies for computer components, it is becoming more and more common the deployment of clusters with hundreds or even thousands of nodes. For such large-size systems, the performance of the file system becomes critical. Traditional systems such as NFS (CALLAGHAN; PAWLOWSKI; STAUBACH, 1995) cannot be used directly since the nature of its centralised server becomes a critical bottleneck for the whole cluster. In other words, scalability is an important issue for file systems of large clusters. In this way, many alternatives to greater scalability, in several directions, are currently being pursued. In this paper we present and compare some of the many current file systems which present extended and/or modified features to accomplish the needed scalability for use with large clusters.

2 File Systems for Beowulf Clusters

The **Network File System** (CALLAGHAN; PAWLOWSKI; STAUBACH, 1995), or NFS, is the *de facto* standard for distributed file sharing in the Unix world, and consequently has been naturally absorbed by the Beowulf cluster model since its first steps (STERLING, 1999).

[1] avila@inf.ufrgs.br Bolsista CAPES
[2] navaux@inf.ufrgs.br
[3] Yves.Denneulin@imag.fr

NFS was developed by Sun Microsystems and first came out in 1985 with the release of SunOS 2.0. It is in fact a protocol for transparent remote access from a client to a server's file system. NFS is designed to be a stateless protocol[4], which means that a crash from one of the clients does not affect the server, and server crashes can be easily and transparently recovered when the server comes back.

The **Berkeley xFS**[5] (ANDERSON, 1995) was a prototype "serverless" file system developed at the University of California at Berkeley from 1993 to 1995. It builds upon several research efforts developed at the time, mainly on RAID, LFS (Log-structured File System), Zebra and Multiprocessor Cache Consistency (all described on the same article from 1995). xFS presents a totally distributed design, where data and control information, or *meta-data*, are spread among the available machines (which may be all or part of the available computing resources) and can dynamically migrate.

A prototype of xFS has been built and tested on a 32-node SPARCStation cluster at Berkeley. Despite the very promising results, however, the project stopped shortly after, and hence, to our knowledge, no further development or porting to Linux has been carried out to the present days.

PVFS, *Parallel Virtual File System* (CARNS, 2000), is a joint project conducted by the Parallel Architecture Research Laboratory, at Clemson University, and the Argonne National Laboratory, both in the USA. The goal of PVFS is to provide a high-performance file system for the Beowulf class of parallel machines, being able to profit from commodity hardware.

The system can be viewed as a simplified version of xFS. Access to permanent storage is performed by *I/O daemons*, and a single, centralised *manager* is responsible for manipulating meta-data. Clients access the file system by means of the PVFS API, which has also been ported to a Linux VFS kernel module (i.e. it can be `mounted`) and to MPI-IO.

NFSP — NFS Parallel — is an extension of the traditional NFS implementation, developed at Laboratory ID/IMAG of Grenoble, that distributes the functionality of the NFS daemon over several processes on the cluster (LOMBARD; DENNEULIN, 2002). The idea behind NFSP is to provide an improvement in performance and scalability and at the same time keep the system simple and fully compatible with standard NFS clients.

Similarly to PVFS, NFSP makes use of I/O daemons to access data on the disks. The role of the NFS server is played by the *nfspd*, defined by the authors as a *meta-server*. This daemon appears to clients as the regular nfsd server; when a request for a given piece of data is received, the meta-server forwards the request to the corresponding I/O daemon, which in turn performs the operation and sends the desired information directly to the client.

Other Approaches — Several other research projects aim at a better performance for cluster file systems, using different approaches. Petal/Frangipani (LEE, 1998) is a parallel file system built upon the concept of a *distributed virtual disk*, where a set of daemons running on a number of machines cooperate to form the view of a single storage device. The *Shared*

[4]The new NFS Version 4 will be stateful

[5]Not to be confused with SGI's XFS, with a capital "X"

Table 1: Overall performance comparison for the analysed systems (bandwidths are presented as an improvement ratio over that of regular NFS)

Item	xFS	PVFS	NFSP
I/O servers	32	24	16
Read bandwidth	11.04	17.76	4.0
Write bandwidth	11.12	18.08	–
Read efficiency	34.75%	74%	25%
Write efficiency	34.5%	75.3%	–

Logical Disk (SHILLNER; FELTEN, 1996) follows the same idea. Another approach is that of *Storage Area Networks*, in which there is a dedicated hardware support for parallel access to permanent storage. Examples are the IBM General Parallel File System (CORBETT et al., 1995), SGI XFS (SWEENEY, 1996), and the OpenGFS (Global File System) (THE..., 2003). Though these systems are of recognised importance, we will concentrate the following analysis on the previous ones, since we are targeted at Beowulf class systems, in which the use of commodity software and components is strongly desired.

3 Analysis of the Presented Systems

Overall Performance This comparison is based on performance measurements, in terms of maximum achievable bandwidth, presented by the authors of each system on the indicated references. It is difficult to compare them directly since the results have been obtained on different test-beds. Therefore, we choose to present, for each system, the results relative to the reported regular NFS performance on the same environment. We also present the efficiency of each system in relation to the number of I/O servers used.

Table 1 summarises the relative performances of each system. We have separated write and read performances since NFSP is not at present optimised for distributed writing, and thus a comparison would be unfair. It is also important to state that the results presented for xFS have been obtained from an early prototype, admittedly reported by the authors as not optimised. The system which presents best overall performance is PVFS, reflecting a very good efficiency in using the available I/O servers and thus reaching improved bandwidth. For both xFS and PVFS, read and write performances are similar, since they use a symmetric approach for both operations, to the difference of NFSP.

Scalability In this item we are evaluating the capacity of each system in increasing performance as the number of clients and I/O servers increase.

NFS scalability, as already exposed, is deficient for large clusters. In fact, this deficiency

depends mostly on the interconnection technology being used. Current off-the-shelf hard disks deliver bandwidths in the range of 50–150 MB/s; this means that access to the NFS server is likely to be limited in about 11 MB/s by an ordinary Fast Ethernet network card even before the disk's full capacity may be reached. In other words, even a small 16-node cluster can be easily affected by the problem.

As a consequence, all of the three systems presented are able to overcome standard NFS performance very quickly, even for a few client nodes; however, they differ in how the performance increase scales. xFS presents good scalability, especially up to about 8 client nodes, when the performance increase is practically linear (around 1 MB/s for 1 client, 7.5 MB/s for 8). NFSP scales linearly up to 6 clients and then stabilises, roughly at 50 MB/s. PVFS, once more, shows very good results: the system has presented practical linear scaling up to 24 client nodes and I/O servers, when 226 MB/s can be achieved.

The limits reached by the three systems reflect the characteristic mentioned before, that performance is limited by network bandwidth rather than that of disk. The last two systems present results obtained using Fast Ethernet as interconnect, where a maximum bandwidth of 9–11 MB/s is usually achieved; as a consequence, 6 NFSP clients are limited at 50 MB/s (8.33 MB/s per client), and 24 PVFS clients are limited at 226 MB/s (9.41 MB/s per client). It also explains why PVFS does not scale beyond this limit: even though up to 30 nodes have been used, the number of I/O servers was kept at 24. In the case of NFSP, the authors suspect that the limiting in performance happens because of the meta-server's CPU saturating; they expect the system to deliver better performance after some optimisation in the code.

These results suggest that the network design in a cluster file system should be carefully studied. Techniques such as channel bonding (using two network cards as one), whose support is readily available in the Linux kernel, or an expected popularisation of Gigabit Ethernet may contribute to that.

Fault Tolerance This feature can be viewed in two levels: temporary service unavailability and crash failure. NFS supports the first, but not the second. Up to version 3, NFS is a stateless protocol, which means that temporary server unavailability is tolerated by the clients, but a crash is fatal. This adapts well to a real scenario, where hardware failures are less frequent but eventual stops/restarts may occur (e.g. when upgrading software "on the fly").

When discussion comes to distributed servers, fault tolerance becomes more difficult do realise or imposes significant overhead, and as such it is frequently left aside. xFS makes use of RAID to provide fault tolerance, as well as for improved performance. Storage servers are divided in groups, and in each group one server is dedicated for parity. Thus, failure in one of the servers (even permanent) can be coped with. NFSP benefits from the same stateless model of NFS, in the sense that temporary failures can be tolerated. The current implementation does not provide a mechanism for tolerating failures in the I/O nodes, but the authors do mention redundancy as a future improvement, which may introduce a level of error recovery capability. The PVFS design does not include support for fault tolerance; this

seems to be planned for version 2 of the system.

Integration with the Beowulf model For integration with the Beowulf model, we consider characteristics such as availability of the software, no requirement for a specific technology, and licensing of the whole system as open source. We also evaluate the complexity of integrating the system in a regular Linux cluster.

Except for xFS, all the other systems fully comply with the first requirements. The former was only implemented for an early version of Solaris, and is to our knowledge not yet available for Linux. NFS is traditionally included on every current Linux distribution. Its use is straightforward, requiring single file configuration on both server and client side. NFSP presents an important feature in this sense, allowing for the client side to remain untouched. Configuration on the server side is also not too complex, being similar to that of NFS.

PVFS represents the most "intrusive" solution, since it requires dedicated configuration on both sides. While the client side does not differ much from NFS, requiring only a kernel VFS (Virtual File System) module to be compiled, the server side does demand further configuration.

4 Final Considerations

Many developments, in several directions, can be observed today concerning high performance file systems for clusters, given that NFS represents a potential bottleneck. In this paper we have concentrated on some of the systems more directly targeted at the Beowulf class of parallel machines, in the sense that no specific hardware or communication technology be required for their proper utilisation. Available commercial file systems for high-performance computing often exhibit that requirement.

Table 2 summarises the analysed features of each system in a simplified form. The Berkeley xFS seems a promising design, since very good results have been obtained, even with an unoptimised prototype; the system, however, has not been further developed or ported to Linux, rendering itself currently not eligible as a solution for Beowulf clusters. PVFS is currently becoming a kind of *de facto* standard in the Beowulf world. The system presents very good performance and scalability, and supports several APIs including a Linux VFS module, with a slight complexity in management. Version 2 is currently being developed, and will include features for grid computing and fault tolerance. NFSP is also in an early stage of development, but presents encouraging results, besides being compatible with the standard NFS client. A design alternative allowing for concurrent write operations would also be desirable.

Table 2: Summary of the analysed features

Item	NFS	xFS	PVFS	NFSP
Performance	regular	good	very good	good
Scalability	poor	good	very good	regular
Fault tolerance	temporary	crash	none	temporary
Beowulf integration	very easy	none	not trivial	easy

References

ANDERSON, T. E. et al. Serverless network file systems. In: ACM. *Proc. of the 15th Symposium on Operating Systems Principles*. Copper Mountain Resort, Colorado, 1995. p. 109–126.

CALLAGHAN, B.; PAWLOWSKI, B.; STAUBACH, P. *NFS Version 3 Protocol Specification: RFC 1831*. [S.l.], jun. 1995.

CARNS, P. H. et al. PVFS: a parallel file system for Linux clusters. In: *Proc. of the 4th Annual Linux Showcase and Conference*. Atlanta, GA: [s.n.], 2000. p. 317–327. Best Paper Award.

CORBETT, P. F. et al. Parallel file systems for the IBM SP computers. *IBM Systems Journal*, v. 34, n. 2, p. 222–248, jan. 1995.

LEE, E. K. et al. *A Comparison of Two Distributed Disk Systems*. [S.l.], abr. 1998.

LOMBARD, P.; DENNEULIN, Y. nfsp: a distributed NFS server for clusters of workstations. In: *Proc. of the 16th International Parallel & Distributed Processing Symposium, IPDPS*. Ft. Lauderdale, Florida, USA: Los Alamitos, IEEE Computer Society, 2002. p. 35. Abstract only, full paper available in CD-ROM.

SHILLNER, R. A.; FELTEN, E. W. *Simplifying Distributed File Systems Using a Shared Logical Disk*. Princeton, NJ, 1996.

STERLING, T. L. *Beowulf Cluster Computing with Linux*. Cambridge: MIT Press, 2002.

STERLING, T. L. et al. *How to Build a Beowulf: a Guide to the Implementation and Application of PC Clusters*. Cambridge: MIT, 1999. 239 p.

SWEENEY, A. et al. Scalability in the XFS file system. In: *Proceedings of the USENIX 1996 Technical Conference*. San Diego, CA, USA: [s.n.], 1996. p. 1–14. Disponível em: <citeseer.nj.nec.com/sweeney96scalability.html>.

THE OpenGFS Project. abr. 2003. Disponível em: <http://opengfs.sourceforge.net>. Access em: abril 2003.