Metaserver Locality and Scalability in a Distributed NFS*

Everton Hermann[1,**], Rafael Ávila[1,***], Philippe Navaux[1], and Yves Denneulin[2]

[1] Instituto de Informática/UFRGS
Caixa Postal 15064
91501-970 Porto Alegre – Brazil
Phone.: +55 (51) 3316-6165; Fax: +55 (51) 3316-7308
{ehermann,avila,navaux}@inf.ufrgs.br
[2] Laboratoire ID/IMAG
51, avenue Jean Kuntzmann
38330 Montbonnot-Saint Martin – France
Phone.: +33 (4) 76 61 20 13; Fax: +33 (4) 76 61 20 99
Yves.Denneulin@imag.fr

Abstract. The leveraging of existing storage space in a cluster is a desirable characteristic of a parallel file system. While undoubtedly an advantage from the point of view of resource management, this possibility may face the administrator with a wide variety of alternatives for configuring the file server, whose optimal layout is not always easy to devise. Given the diversity of parameters such as the number of processors on each node and the capacity and topology of the network, decisions regarding the locality of server components like metadata servers and I/O servers have a direct impact on performance and scalability. In this paper, we explore the capabilities of the dNFSp file system on a large cluster installation, observing how scalable the system behaves in different scenarios and comparing it to a dedicated parallel file system. Our obtained results show that the design of dNFSp allows for a scalable and resource-saving configuration for clusters with a large number of nodes.

Topics: Cluster and grid computing, parallel I/O, parallel and distributed computing.

1 Introduction

Solutions for efficient management of I/O in large clusters have long been the focus of several research groups and industrials working on parallel computing [1]. Ranging from RAID arrays and fibre optics to virtual distributed disks, many approaches have been proposed in the last decade that vary considerably in terms of performance, scalability and cost.

* Candidate to the best student paper award.
** Work partially supported by CAPES and CNPq.
*** Work supported by HP Brazil grant.

M. Daydé et al. (Eds.): VECPAR 2006, LNCS 4395, pp. 145–157, 2007.

In previous works [2,3], we have presented the *dNFSp* file system, an extension of NFSv2 that aims at improving both performance and scalability of a regular NFS server while keeping its standard administration procedures and, mainly, compatibility with the regular NFS clients available on every Unix system. Similarly to other parallel file systems such as PVFS [4] and Lustre [5], dNFSp is based on a distributed approach where the gain in performance is obtained by executing tasks in parallel over several machines of the cluster.

One important aspect of a parallel file system is its capability of leveraging existing resources. In the case of commodity clusters, the hard disks that are installed on the compute nodes are frequently under-used: a typical GNU/Linux node installation takes only a few gigabytes, and today's PCs are hardly ever configured with less than 40 gigabytes of storage. This leaves us with at least 75% of the total hard disk capacity available for the storage of data, and consequently it is important that a cluster file system have the ability to use it.

dNFSp provides such a feature, so that the storage on the compute nodes can be used to form a single cluster file system. It is then up to the cluster administrator to decide how to configure the system, finding a good balance between resource utilization, performance and scalability, which might not be an obvious task.

For this reason, we have conducted a series of experiments varying the configuration of dNFSp on a large cluster. This allowed us to watch how scalable the system os. Also it was possible to identify layouts best suited for one or another situation, and whose results and conclusions are presented in this work.

In the remainder of the paper, Section 2 presents the dNFSp file system and its main characteristics, with the purpose of providing some background knowledge; Section 3 describes in more details the experiments we have conducted and introduces the evaluation criteria; in Section 4 we present the results obtained in the experiments and provide the discussion which is the focus of this work; Section 5 brings a comparison of our work to systems with related objectives, and finally Section 6 draws some conclusions on the obtained results and analysis and reveals future directions.

2 dNFSp – A Distributed NFS Server

The NFSp project [6] has been established in 2000 at the *Laboratoire Informatique et Distribution* of Grenoble, France, with the goal of improving performance and scalability in a regular NFS installation. The main idea of the project is to provide a cluster file system that benefits from the standard administration procedures and behavior of a well-known protocol such as NFS. As a result, NFSp — for *parallel* NFS — presents some simple extensions to the NFS server implementation that distributes its funcionalities over a set of nodes in the cluster, thus gaining performance. On the other hand, the client machines do not have to be modified at all, favoring portability.

As a subproject within the NFSp group, dNFSp has been proposed as a further extension to the model, aiming at an improvement on concurrent write operations by client machines.

Figure 1 depicts the distribution model proposed by dNFSp. The top of the figure shows the two main server components: the *I/O daemons*, or IODs, and the *metaservers*. The metaservers are daemons that play the role of the NFS server, serving clients' requests and cooperating with each other to form the notion of a single file server. The IODs work as backends for the metaservers, being responsible solely for data storage and retrieval. On the lower part of the figure, client machines connect to the metaservers in the same way that clients connect to a regular NFS server.

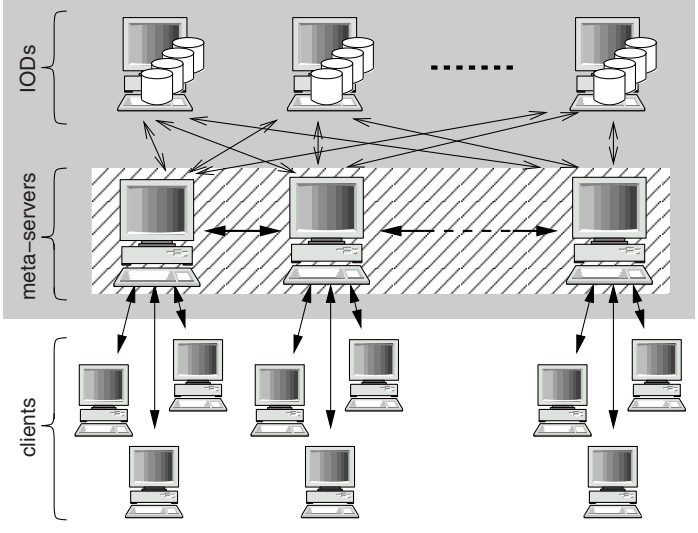

Fig. 1. Distributed metaserver architecture of dNFSp

Each client is connected to one metaserver, which is responsible for handling its requests. Operations involving only metadata are replied directly by the metaserver, which in some cases can contact other metaservers to obtain the needed metadata. I/O operations are forwarded by the metaserver to the IODs, which will perform the operation and reply directly to the client. In the case of read operations, the file contents are transferred directly from the IODs to the clients, allowing parallel reads up to the number of available IODs. In the case of write operations, the data are transferred from the client to the metaserver, and then forwarded to the IODs. Therefore, global write performance depends both on the number of IODs and the number of metaservers.

The metaservers exchange information to keep metadata coherence across all metaservers. The information is retrieved only when needed by a client, avoiding unnecessary network traffic. However, there are situations when all metaservers must be contacted (e.g. *lookup()* upon file creation), and this communication becomes more visible as we increase the number of metaservers.

The extensions introduced by dNFSp have been implemented on an existing NFSp prototype, and a performance evaluation has been carried out previously [3] with the execution of some micro benchmarks. As an illustration of dNFSp raw performance, Figure 2 shows the results obtained for concurrent read and write operations (each client reads/writes one independent 1 GB file) in comparison to the performance of a regular NFS server using Fast Ethernet network (∼11 MB/s). As expected, one obtains an increased throughput when more than one client read/write at the same time.

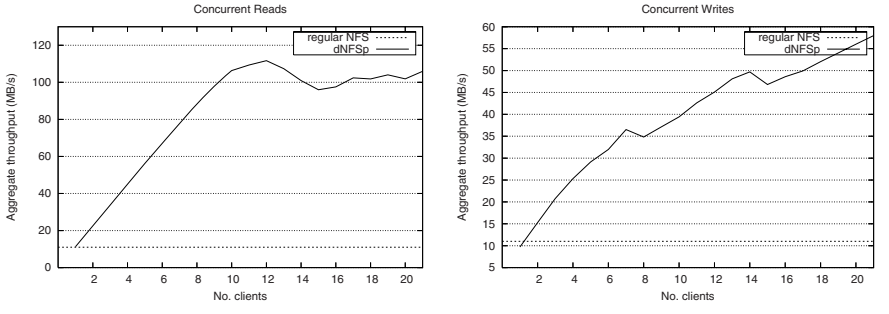

Fig. 2. dNFSp performance for concurrent read and write operations using 12 IODs and 7 metaservers

3 Benchmark and Cluster Environment

The measurements we carried out have the objective of evaluating the scalability of dNFSp in a large number of nodes using a real application-based benchmark. The benchmark we used is the *NAS/BTIO*, and the machine used to run the applications is the *INRIA i-cluster2*[1]. Both the application and the cluster are detailed in the following sections.

3.1 The NAS/BTIO Benchmark

The BTIO Benchmark is a part of the NAS Parallel Benchmarks (NPB) [7]. It is commonly used for evaluating the storage performance of parallel and distributed computer systems. The application used by BTIO is an extension of the BT benchmark [8]. The BT benchmark is based on a Computational Fluid Dynamics (CFD) code that uses an implicit algorithm to solve the 3D compressible Navier-Stokes equations. BTIO uses the same computational method employed by BT. The I/O operations have been added by forcing the writing of results to disk. In BTIO, the results must be written to disk at every fifth step of BT.

The number of process running one execution of BTIO must be a perfect square (1, 4, 9, 16, etc.). The problem size is chosen by specifying a class. Each

[1] http://ita.imag.fr

class corresponds to the dimensions of a cubic matrix to be solved by the application: class A (64^3), class B (102^3), class C (162^3). We have chosen class A since it was enough to have a good mixing of computation and file system operations. Moreover, using a larger class has not changed the profile of the results.

Another customization of BTIO is the way the I/O operations are requested to the file system. There are four flavors that can be chosen at compilation time:

- BTIO-full-mpiio: This version uses MPI-IO file operations with *collective buffering*, which means that data blocks are potentially re-ordered previously to being written to disk, resulting in coarser write granularity
- BTIO-simple-mpiio: Also uses MPI-IO operations, but no data re-ordering is performed, resulting in a high number of seeks when storing information on the file system
- BTIO-fortran-direct: This version is similar to simple-mpiio, but uses the Fortran direct access method instead of MPI-IO
- BTIO-epio: In this version, each node writes in a separate file. This test gives the optimal write performance that can be obtained, because the file is not shared by all the processes, so there is no lock restriction. In order to compare with other versions, the time to merge the files must be computed, as required by the Application I/O benchmark specification.

In order to perform MPI-IO operations using an NFS-based file system, it would be necessary to have an implementation of the NFSv3 protocol, due to the need of controlling file access by means of locks. Since dNFSp was implemented based on the NFSv2 protocol, it has no lock manager; hence, we decided to use the BTIO-epio version of the benchmark. Furthermore, using epio allows one to achieve optimal performance results, since the data are written to individual files by each node.

With the goal of having a more write-intensive benchmark, we have made a small change in the BTIO benchmark code, modifying the frequency of file writes. The original code performes writes on every five iterations, resulting in a total amount of writes of 400 megabytes; with the modification, BTIO performs writes on every iteration, resulting in 2 gigabytes of written data. The results of each computing step are appended to the end of the file used by the process. The granularity of writes changes with the number of nodes used on the computation. Changing the frequency of writings allowed us to make the differences between the file systems more visible.

3.2 The i-Cluster2

The *i-cluster2* [9] is installed in Montbonnot Saint Martin, France, in the INRIA Rhône-Alpes facility. The cluster is composed by 100 nodes. Each node is equipped with a dual Itanium2 900 MHz with 3 gigabytes of memory and a disk storage with 72 gigabytes, 10000 rpm, SCSI. All the nodes are interconnected using a 1 Gigabit Ethernet network, Fast Ethernet network and Myrinet Network. The experiments were performed using the 1 Gigabit Ethernet network.

The software installed on i-cluster2 is based on Red Hat Enterprise Linux AS release 3 distribution, with a Linux kernel version 2.4.21. The MPI implementation used with the BTIO benchmark is mpich version 1.2.6.

4 Performance, Scalability and Locality Evaluation

In this section we present the results obtained with our experiments. The reported execution times are those informed by BTIO at the end of the execution, together with a confirmation of correct computation. Each value reported is the arithmetic mean of at least 5 runs of BTIO with the same configuration, so as to obtain a stable value. Standard deviations lie within a maximum value of 3 seconds.

4.1 Performance Analysis

The first step in our analysis of dNFSp has been an evaluation of the performance of the system on the i-cluster2. We have run BTIO on a large subset of the available nodes, and compared the performance of dNFSp with that of a dedicated parallel file system. We have chosen PVFS [4] for this task, as it is a representative parallel file system in the Beowulf cluster context in which our work is inserted. It wasn't possible to perform tests using Lustre because it needs a kernel patch to the system to run, and we didn't have the permission needed to do this task.

For both systems, we have varied the number of IODs in the file server from 4 up to 12 IODs, in steps of 2. In the case of dNFSp, we always use a number of metaservers equal to that of IODs. PVFS uses only one extra node in all cases, for the *manager*. The experiments have been executed from 4 up to 49 clients, respecting the feature of BTIO that the number of clients must be a perfect square.

We show, in Figure 3, the results obtained using the minimum and the maximum number of IODs, respectively 4 and 12. Intermediate configurations have shown proportional variation.

When the file system is accessed by a small number of clients, a shorter number of metaservers has shown better performance results, because a higher number of metaservers results in more management communication. In this case, the application does not have enough nodes to benefit from all the parallelism offered by the file system.

As expected, execution times drop as the number of clients increase. For dNFSp, the reduction in execution time is progressive on the whole range of clients, except for the case of 49 clients using 4 IODs, where it slightly starts to rise again. For PVFS, one observes a lower limit at around 100 seconds. We conclude that the main cause for the limitation in both systems is that, as the number of clients increase, the amount of data written by each one decreases, and reaches a point where parallelism does not pay off anymore due to the management cost of the striping mechanism. dNFSp seems to handle the situation better than PVFS, reaching around 47 seconds in the case of 12 IODs.

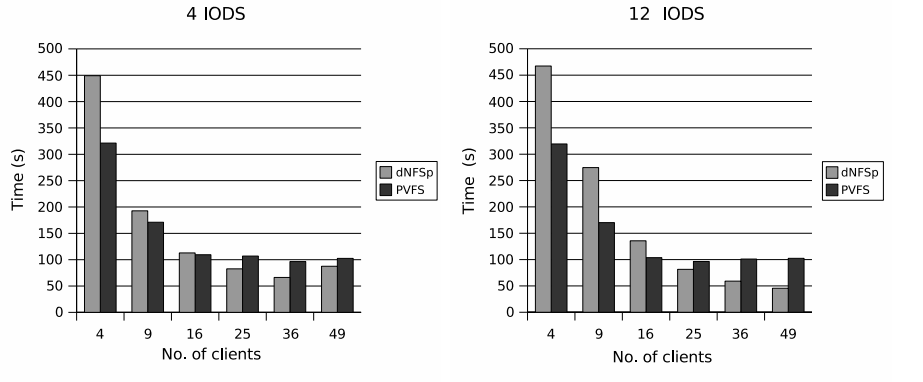

Fig. 3. Performance comparison for dNFSp vs. PVFS using 4 and 12 IODs on the file server

It is important to remark that such experiments were run using PVFS v1 (more precisely, version 1.6.3), while PVFS2 is already available and should presumably yield more performance than its predecessor. PVFS2 was effectively our initial choice for comparison, not only because of its performance, but also because it features a distributed metadata model which is closer to that of dNFSp. However, early experiments with that version on the i-cluster2 resulted in strangely poor performance (results are shown in Table 1 for information). One reason to this poor result is the writing profile of BTIO, which appends contents to the end of the files, changing metadata information on each write. The relaxed model used by dNFSp does not require updating metadata on all the metaservers so there is no extra communication when an append is performed. While we are still investigating further causes for that behavior, we chose to report results for PVFS v1, which nevertheless represents no loss in significance since it is still fully supported by the developers as a production system.

4.2 Scalability Evaluation

As a subsequent analysis of our experiments we have compared how both file systems react to the addition of more nodes to the file system. The number of IODs, metaservers and clients is the same as described in the previous section.

In Figure 4 we have three samples from our experiment. In the first chart we have kept the number of BTIO clients on 4 and varied the number of IODs and metaservers. We can see that both file systems sustain an almost constant performance due to the fact that the application doesn't have enough clients to stress the capability of storage offered by the file systems. In this case, dNFSp even shows a small decrease in performance as we add more nodes. This loss of performance comes from the metaserver communication, which is more significant when we have more of them. The better performance of PVFS lies on the size of messages. As we have only four BTIO clients, the amount of computation designated to each node is large resulting in larger writes on the file system.

Table 1. Sample execution times (in seconds) obtained for dNFSp and PVFS v1 in comparison to PVFS2

No. of clients	dNFSp	PVFS v1	PVFS2
4	464.416	319.273	363.395
9	272.728	170.663	263.655
16	135.164	103.247	253.88
25	76.712	95.35	252.505
36	53.745	96.313	293.44
49	43.776	105.44	353.64

The second chart shows the transition case where both file systems have a similar behavior. Using 25 clients BTIO seems to have a block size that results in similar performance to both file systems. They have an improvement of performance as we add more nodes to the file system, reaching a limit where adding more nodes increases the execution time instead of reducing.

In the third chart we show a more stressing case, where we have 49 clients accessing the file system. In this situation we can see that from 4 to 10 nodes dNFSp has an improvement of performance as we increase the file system size. When we reach 12 IODs and 12 metaservers, the overhead added by the insertion of more nodes is not compensated by the performance gain. The clients don't have enough writes to stress the file system, and the communication between metaservers is more expressive, falling in the same situation shown by the four clients sample. PVFS has shown a performance limitation when we have small writes to the file system. As the number of clients is larger than the previous case, we have smalls chunks of data being written.

4.3 Metaservers and IODs Locality Impact

As a last experiment, we have investigated the capabilities of dNFSp in saving cluster nodes for the deployment of the file server. As usual in distributed file systems (and distributed systems in general) like dNFSp and PVFS, each task of the system is usually performed on a distinct machine or compute node. For example, in the previous experiments we have always used distinct nodes for running the IODs and the metaservers/manager.

It would be desirable, however, that one could make use of as few nodes for the file server as possible, in order to maximize the number of nodes available for the real computing tasks. While the decision depends mostly on the amount of storage desired for the server, some considerations can be made regarding performance and scalability that might allow for a shorter number of nodes than that initially accounted. This is specially true if the compute nodes are dual-processed.

The results in Figure 5 correspond to the execution of BTIO with dNFSp with IODs and metaservers running on the same nodes, compared to the original execution where the two entities run on separate nodes. Again, we show results for a small and a large number of client nodes.

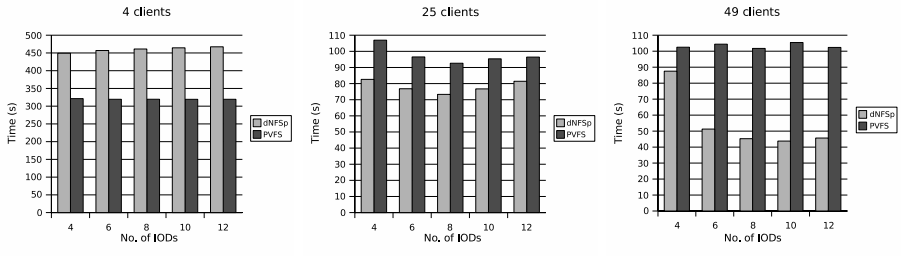

Fig. 4. Scalability comparison for dNFSp vs. PVFS using 4, 25 and 49 BTIO clients processes

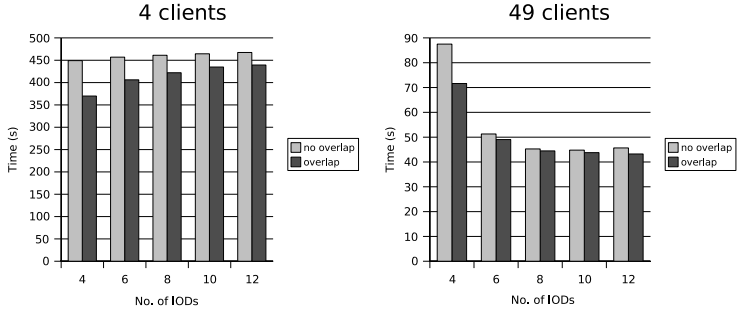

Fig. 5. Results for dNFSp with and without overlapping IODs and metaservers on the same nodes

Two different situations are presented, both favoring the overlapping of IODs and metaservers. In the first case, with few client nodes, more information is written by each single client, and thus there is a visible difference in performance in favor of the overlapping configuration, since communication between the IOD and the metaserver on the same node is done faster (by means of memory copy). Approximately $1/N$ requests, where N is the number of IODs, can be processed locally, without the need of contacting an IOD through the network. On the second case, there are much more, smaller client writes, and consequently the differences are not much evident. Increasing the number of IODs also contributes to minimize the differences, as the probability of performing a request locally decreases as the number of striping slices grows. As a conclusion, we can see that such an overlapping configuration can be employed without loss of performance, contributing to the amount of nodes dedicated to computation.

5 Related Work

The increasing performance gap between I/O and processor has placed the file system performance as the most severe bottleneck to applications that massively use the storage subsystem [1,10]. Several approaches have been proposed since

the deployment of the first large-scale parallel machines. Many are based on the use of specialized technologies (e.g. RAID, fiber optics) as a means to increase performance, such as GPFS [11] and GFS [12]. This kind of system usually relies on the concept of a Storage Area Network (SAN), which basically defines a common storage "device" composed of several physical devices. As such, scalability is a direct consequence of this concept. Other projects like Petal/Frangipani [13,14] and the Shared Logical Disk [15] make use of the same concept, but the SAN is implemented in software over a network. Good performance and scalability thus depend heavily on the communication technology.

Research projects like PVFS [4] and Lustre [16] follow another trend. To achieve high performance on I/O operations, these file systems distribute the functions of a file system across a set of nodes in a cluster. To perform parallel I/O operations, they stripe the data across the nodes, keeping the striping transparent to the application.

PVFS is a parallel cluster file system composed of two types of nodes: the *I/O server* and the *manager*, which is a metadata server. The nodes in the cluster used by the file system can be configured as I/O servers, and one of them as a manager. Lustre is an object-based file system designed to provide performance, availability and scalability in distributed systems. Like PVFS, Lustre comprises two types of server nodes: Metadata Servers (MDS) are responsible for managing the file system's directory layout, as well as permissions and other file attributes. The data is stored and transferred through the Object Storage Targets (OST). Figure 6 shows the architecture of PVFS and Lustre in comparison to that of dNFSp.

High performance in PVFS is achieved by distributing the contents of a file across the I/O server nodes (striping). Clients are allowed to access the contents of the file in parallel. The way the files are striped is handled by the metadata manager, which is also responsible for managing file properties and a name space to applications, but has no participation in I/O operations. The I/O servers are accessed directly by the clients to deal with the data transfers. The user can access the file system through the PVFS library or using an OS-specific kernel module. The latter allows the user to mount the file system using a POSIX interface, and access files as any other file system.

In Lustre, similarly to PVFS, the client contacts the Metadata Servers to know which OST contains a given part of a file. After obtaining this information, the client establishes direct connections to the OST performing reads and writes. The MDS has no intervention in the I/O process, being contacted by the OST only to change file attributes like file size. Both types of nodes can have replicas working in pairs and taking the place of each other when a failure occurs. Figure 6 shows an *active* MDS, and its replica is represented by the *failover* node. Information concerning the overall system configuration is stored in a Lightweight Directory Access Protocol (LDAP) server. In the event of a MDS failure, the client can access the LDAP server to ask for an available MDS.

As in dNFSp, PVFS version 2, or PVFS2 [17], has the option of running more than one manager. The main difference lies on the way PVFS controls the

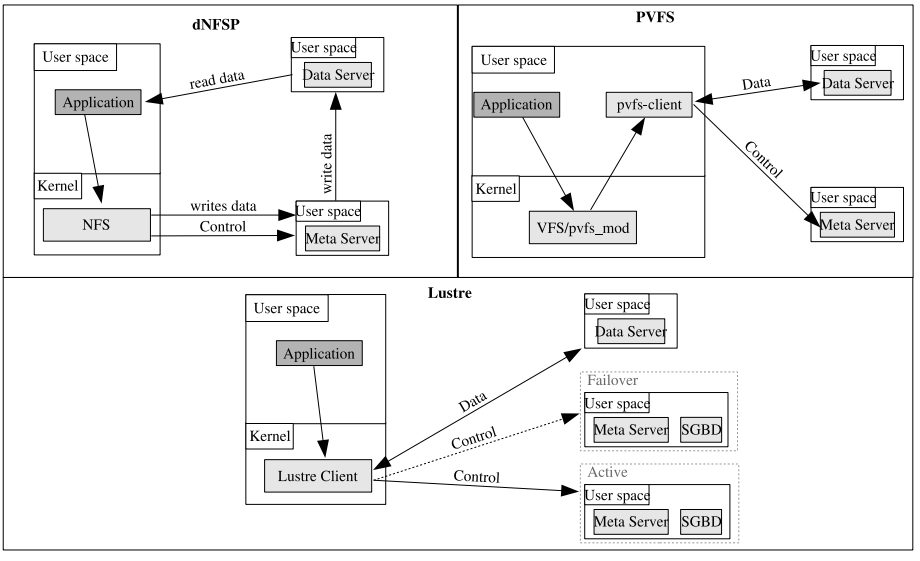

Fig. 6. Architecture of the related cluster file systems

distribution of metadata. Each manager stores the metadata information about a range of files, while in dNFSp each server has the metadata information about all the files. The PVFS approach can result in a surcharged manager when all the clients access files in the same range.

6 Conclusions and Final Considerations

The execution of the BTIO benchmark with dNFSp and PVFS on the i-cluster2 has confirmed the objectives of our system in providing good performance and scalability while keeping compatibility with NFS. The results show very good, scalable performance in comparison to a dedicated parallel file system. dNFSp is able to reach the same level of performance of PVFS, and many times even reach beyond it. We understand that this advantage comes from the fact that dNFSp can tolerate a smaller size of writes than PVFS before reaching the point where parallelism in no longer favorable. When configured with a large number of clients, dNFSp has outperformed PVFS in up to 50% of its execution time.

Another positive aspect of our benchmarking is that dNFSp performs efficiently when IODs and metaservers have been run together. The performance results obtained using IODs and metaservers on the same node were up to 17% faster than in the case where metaservers and IODs were run on distinct machines. This allows for a resource-saving configuration which maximizes the availability of compute nodes without sacrificing performance. Although the nodes of the i-cluster2 are dual-processed, which favors this configuration, we believe that a similar approach, at least partial, should be possible on single-processor

clusters, since the IODs present a typical I/O-bound profile, while the metaservers do little disk activity. This evaluation was not possible on the i-cluster2, as it would require booting with a non SMP-enabled kernel. We intend to carry it out in the next stage of the project.

One of the future activities in dNFSp is the implementation of a dedicated communication mechanism between metaservers, given some loss of performance in a few of the experiments due to the heavy lookup mechanism that the metaservers perform. The dedicated protocol should minimize the impact of lookup operations by implementing some kind of prefetching and message aggregation for the exchange of metadata. Also, we are planning a porting of dNFSp to NFS version 3. This will allow us to profit from the changes on the protocol, like asynchronous I/O operations and the larger block size limit. Another aspect to be worked upon is fault tolerance and replication, which are being studied and shall be included in the upcoming versions.

References

1. Schikuta, E., Stockinger, H.: Parallel I/O for clusters: Methodologies and systems. In Buyya, R., ed.: High Performance Cluster Computing: Architectures and Systems. Prentice Hall PTR, Upper Saddle River (1999) 439–462
2. Kassick, R., Machado, C., Hermann, E., Ávila, R., Navaux, P., Denneulin, Y.: Evaluating the performance of the dNFSP file system. In: Proc. of the 5th IEEE International Symposium on Cluster Computing and the Grid, CCGrid, Cardiff, UK, Los Alamitos, IEEE Computer Society Press (2005)
3. Ávila, R.B., Navaux, P.O.A., Lombard, P., Lebre, A., Denneulin, Y.: Performance evaluation of a prototype distributed NFS server. In Gaudiot, J.L., Pilla, M.L., Navaux, P.O.A., Song, S.W., eds.: Proceedings of the 16th Symposium on Computer Architecture and High-Performance Computing, Foz do Iguaçu, Brazil, Washington, IEEE (2004) 100–105
4. Carns, P.H., Ligon III, W.B., Ross, R.B., Thakur, R.: PVFS: a parallel file system for Linux clusters. In: Proc. of the 4th Annual Linux Showcase and Conference, Atlanta, GA (2000) 317–327 Best Paper Award.
5. Cluster File Systems, Inc.: Lustre: A scalable, high-performance file system (2002) Available at http://www.lustre.org/docs/whitepaper.pdf (July 2004)
6. Nfsp: homepage (2000) Available at <http://www-id.imag.fr/Logiciels/NFSP> Access in May 2005.
7. Wong, P., der Wijngaart, R.F.V.: NAS Parallel Benchmarks I/O Version 2.4. RNR 03-002, NASA Ames Research Center (2003)
8. Bailey, D.H., et al.: The NAS parallel benchmarks. International Journal of Supercomputer Applications **5**(3) (1991) 63–73
9. i-Cluster 2 (2005) Available at <http://i-cluster2.inrialpes.fr> Access in May 2005.
10. Baker, M., ed.: Cluster Computing White Paper. IEEE Task Force in Cluster Computing (2000) Available at http://www.dcs.port.ac.uk/∼mab/tfcc/WhitePaper/final-paper.pdf Final Release, Version 2.0.
11. Schmuck, F., Haskin, R.: GPFS: A shared-disk file system for large computing clusters. In: Proc. of the Conference on File and Storage Technologies, Monterey, CA (2002) 231–244
12. The openGFS project (2003) http://opengfs.sourceforge.net

13. Lee, E.K., Thekkath, C.A.: Petal: Distributed virtual disks. In: Proc. of the 17th International Conference on Architectural Support for Programming Languages and Operating Systems, Cambridge, MA (1996) 84–92
14. Thekkath, C.A., Mann, T., Lee, E.K.: Frangipani: A scalable distributed file system. In: Proceedings of the 16th ACM Symposium on Operating Systems Principles, Saint Malo, France, New York, ACM Press (1997) 224–237
15. Shillner, R.A., Felten, E.W.: Simplifying distributed file systems using a shared logical disk. Technical Report TR-524-96, Dept. of Computer Science, Princeton University, Princeton, NJ (1996)
16. Schwan, P.: Lustre: Building a file system for 1000-node clusters. In: Proceedings of the 2003 Linux Symposium. (2003)
17. Latham, R., Miller, N., Ross, R., Carns, P.: A next-generation parallel file system for Linux clusters. LinuxWorld Magazine (2004)