

Improving the Performances of a Distributed NFS Implementation

Pierre Lombard, Yves Denneulin, Olivier Valentin, Adrien Lebre

▶ **To cite this version:**

Pierre Lombard, Yves Denneulin, Olivier Valentin, Adrien Lebre. Improving the Performances of a Distributed NFS Implementation. PPAM 2003: Parallel Processing and Applied Mathematics, Sep 2003, Czestochowa, Poland. pp.405-412, 10.1007/978-3-540-24669-5_53 . hal-01664538

HAL Id: hal-01664538

https://inria.hal.science/hal-01664538

Submitted on 14 Dec 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Improving the Performances of a Distributed NFS Implementation*

Pierre Lombard, Yves Denneulin, Olivier Valentin, and Adrien Lebre

Laboratoire Informatique et Distribution-IMAG
51 avenue J. Kuntzmann, 38 330 Montbonnot Saint-Martin, France
{plombard,denneuli,ovalenti,lebre}@imag.fr

Abstract. Our NFS implementation, NFSP (*NFS Parallèle*) aims at providing some transparent ways to aggregate unused disk space by means of dividing a usually centralized NFS server into smaller entities: a metadata server and I/O servers. This paper illustrates the issues related to increasing the performances of such an implementation. Two different approaches have been taken: distributing the load across several servers and implementing the server in a more efficient and intrusive way (in kernel mode). The results obtained with both versions are given and compared to the ones of the first user-mode implementation.

1 Introduction

Today's low-cost clusters are often built by using off-the-shelf hardware: each node has its own storage capability, usually only used to store the operating system and the runtime environment. As the hard disk capacity increases, most of the disk space of the nodes remains unused but for temporary files since the users prefer having their files available on every nodes. Partial solutions imply investing in an expensive storage architecture (SAN or RAID servers), yet the disk space is still wasted on the disks of the nodes. Systems providing an aggregation of the unused disk space and the existing ones often implement new protocols or file system types, which may not be considered as a seamless integration for the clients.

Such issues try to be solved by the NFSP project. When the NFSP project was started in mid 2001[1], we chose to use standard and well defined protocols to implement a new kind of NFS server. The first prototype implemented was based on the Linux user-mode server. The first experimental results we got with this implementation highlighted the cost of running the daemon in user-mode. To improve this we tried two methods: balancing the load between several servers and making a more efficient implementation of the server itself. This paper presents these two approaches and compares them from a performance point of view. After this introduction, some related works in the distributed file systems field are

* This work is a part of the research project named "APACHE" which is supported by CNRS, INPG, INRIA and UJF. Some resources were provided by the ID/HP *i-cluster* (More information is available at http://icluster.imag.fr/)

R. Wyrzykowski et al. (Eds.): PPAM 2003, LNCS 3019, pp. 405–412, 2004.
© Springer-Verlag Berlin Heidelberg 2004

shown in section 2. Then the NFSP principles are explained in section 3 and the two methods for improving performances are detailed in sections 4 and 5 which contain performances evaluation. Eventually, some future tracks of research will be tackled in section 6.

2 Related Works

A large amount of work has been carried out in the network file system since the 1980s. Among the first ones, still used nowadays are Sun NFS and Carneggie Mellon's AFS. NFS is aimed at sharing files among nodes in the same LAN whereas AFS is more suited for WAN architecture. A NFS [2,3] server is made of a node exporting its local file system to the clients who access it through a remote mounting operation. NFS is a stateless protocol, no state is kept on the server side so every operation is self sufficient. This gives NFS some protection against temporary faults. However since the access point is unique for all clients the implementation is inherently centralized and so the storage space is limited to the one on the server. This is not the case for AFS which is a fully distributed file system: servers across different sites cooperate to share the same space and offer all the data they contain to their clients which use as a mounting point a server node part of the global architecture. Contrary to NFS, AFS is a stateful system and so coherency is different from the one found in NFS: when a node opens a file a memory of this operation is kept on the server so when another node access the same file for a write operation a cache invalidation message is sent to all the nodes who opened it. However, this strong coherency implies in high cost in terms of network latency, and thus requires a fast network.

In both cases, the goal of these systems is to provide shared storage for users, which is usually different from the needs of current cluster workloads. Indeed, the development of scientific applications has incurred in new constraints (huge amount of data, level of coherency, fine-grained sharing) on the previous file systems, which led to the design of new storage systems.

A first group of solutions, in order to meet the above needs, might be seen as hardware-based.

File systems such as Sistina's GFS[4] and IBM's GPFS[5] are thought for specialized SAN architectures. Both systems have their data and metadata distributed across the SAN and offer advanced locking and sharing facilities of files. However, the performances of such a system is intimately related to the performances of the storage system underneath. For instance, the GFS handling of coherency relies on an extended SCSI instruction sets. As for GPFS, providing things such as fine-grained coherency by means of software requires a fast and low-latency network like those of the SAN's. Another quite promising new system, LUSTRE, being developed since 2000[6,7] by ClusterFS Inc. aims at satisfying huge storage and transfers requirements as well as offering a Posix semantics. To achieve these goals, clients, meta-data servers (MDS) and object storage targets(OST)[1] are connected by means of a fast network.

[1] Some kind of specialized smart storage.