# I/O Data Mapping in *ParFiSys*: Support for High-Performance I/O in Parallel and Distributed Systems

J. Carretero and F.Pérez and P. de Miguel and F. García and L. Alonso

Universidad Politécnica de Madrid (UPM)
e-28660 Madrid, España, E-mail: jcarrete@fi.upm.es

**Abstract.** This paper gives an overview of the I/O data mapping mechanisms of *ParFiSys*. Grouped management and parallelization are presented as relevant features. I/O data mapping mechanisms of *ParFiSys*, including all levels of the hierarchy, are described in this paper.

## 1 Introduction

MPPs, distributed memory systems with a high number of processors, provide a new I/O system, relying on parallel hardware. Their I/O subsystem architecture, which is distributed in nature, usually consists on several independent I/O nodes supporting one or more secondary storage devices. Using parallel I/O systems and parallel file systems seems to be a good approach to take advantage of the inherent parallelism of the MPP, as shown in some parallel file systems as Vesta [3]. In this paper we give an overview of some aspects of *ParFiSys* [2], a parallel file system developed at the UPM to provide I/O services for the GPMIMD machine, an MPP developed within the ESPRIT program P-5404. The main design goals of *ParFiSys* were to provide I/O services to scientific applications requiring high I/O bandwidth, to minimize application porting effort, and to exploit the parallelism of generic message-passing multicomputers, including processing nodes (PNs) and I/O nodes (IONs). To fully exploit all the parallel features of the I/O hardware, the architecture of *ParFiSys* is clearly divided in two levels: file services and block services. The first level is comprised into a component named *ParClient*. The second architectural level, named *ParServer* and located at the ION, deals with logical block requests interacting directly with the I/O devices located on its own ION.

## 2 Data Distribution in *ParFiSys*

To enhance flexibility, *ParFiSys* mapping is based on a very generic **distributed partition** represented as the tuple $(\{NODE_n\}, \{CTLR_c\}_n, \{DEV_d\}_{nc})$, which describes the set of I/O nodes, controllers per node, and devices per controller of the partition. The current implementation of *ParFiSys* supports three kinds of predefined file systems (figure 1): UNIX-like, distributed extended, and distributed cyclic. To reduce contention in ParServers and to increase fault tolerance, the distribution is applied to data and metadata of the file system. A distributed file system has a replicated superblock, and distributed bitmaps and i-node blocks. Having replicated superblocks allows *ParFiSys* to *mount* file systems with some devices damaged. Distributing bitmaps and i-nodes

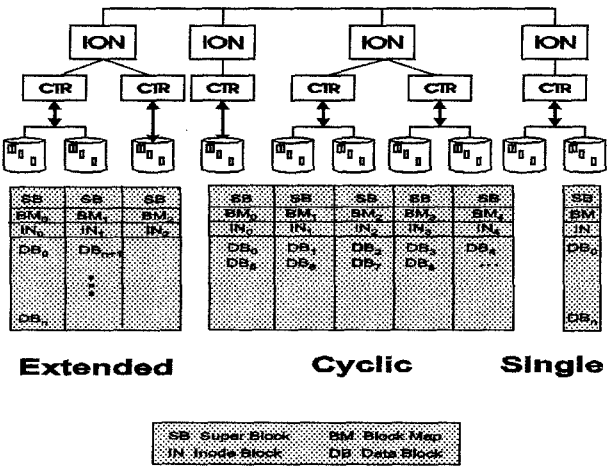


**Fig. 1.** *ParFiSys* Distributed File Systems

increase *ParFiSys* performance by balancing the load due to metadata management among all the ParServers of the file system.

## 3 Data Mapping in *ParFiSys*

Usually the user vision of the file is a byte stream, whereas the file system vision is a set of scattered logical blocks. Moreover, in parallel and distributed file systems, the blocks may be spread out among several ION and devices [1]. Thus, a parallel file system as *ParFiSys* must be able to establish some correspondence between the user and the physical image of data. To satisfy user I/O requests, each ParClient and ParServer must have some knowledge of where the data corresponding to the user image are located. Mapping functions have to be established from the user data structure to the relevant ION, controllers, devices, and disk blocks. The approach suggested for the NCube's I/O software [4] has been followed in *ParFiSys*, where data mapping is solved stepping through four correspondences: File Block to Byte (FBB), File Block to I/O Node (FBION), File Block to Controller (FBC), and File Block to Device (FBD). The two uppermost levels (FBB and FBION) are accomplished at the ParClients. The two lowermost levels are provided by the ParServers. A complete translation to *write* data is defined as $FBB^{-1}$ *o* (*FBION o FBC o FBD*)

FBB is only related to the file image used in *ParFiSys*, a UNIX-like image viewed as a string of bytes. High level I/O operations executed in each ParClient (read, write, ...) translate user I/O requests, defined in bytes, to file system block ones. The block-based approach, used on most file systems, has several drawbacks in high-performance I/O systems, where I/O requests are usually large: cache management cost is proportional to the number of blocks, there is no support to manage several user requests as a single one, and a remote access to a ParServer may be required for each requested block. To enhance FBB correspondence the whole user buffer is mapped to blocks on a single operation in *ParFiSys*, which highly reduces the number of accesses to indirect blocks. In block-based file systems, the number of accesses ($na$) is linear to the depth

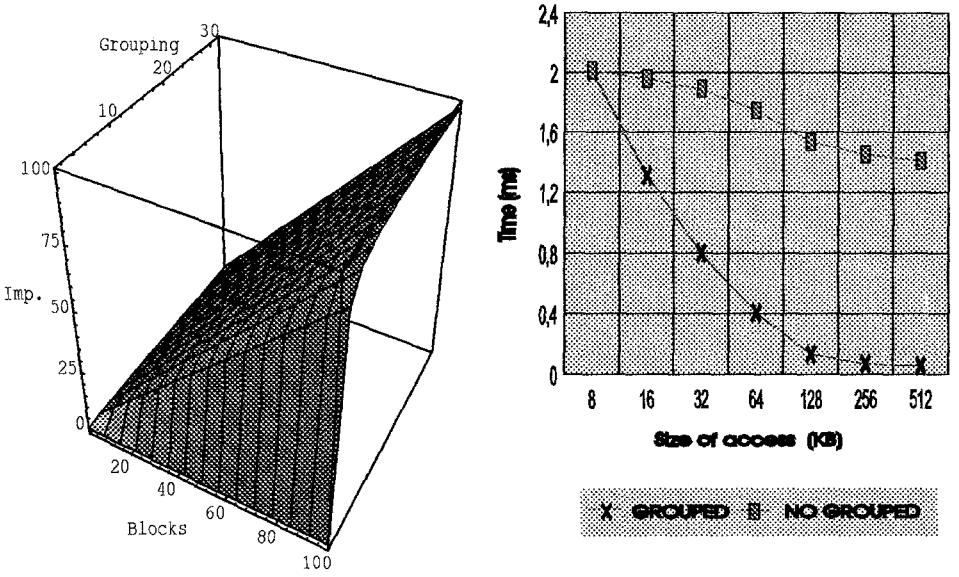


**Fig. 2.** FBB Improvement in ParFiSys

of the indirection level used (single, double, triple). The FBB correspondence used in *ParFiSys* allows to obtain a lists of $n$ blocks corresponding to the whole user request with a single mapping operation. To avoid problems due to greedy processes executing many large I/O requests, a uppermost limit (*FBB factor*) is defined for the size of the user buffer that can be mapped on a single operation. It is computed using a configuration parameter (*FBB maximum*) affected by a weight factor, which depends on the process I/O behavior. The number of accesses ($na$) is now reduced proportionally to the *FBB factor* ($k$), as reflected below:

$$na = \sum_{j=1}^{i} (\frac{n}{k^j} + 1) * j \tag{1}$$

where $n$ is the number of blocks, and $i$ is the indirection level. The improvement achieved with the *ParFiSys* approach is shown in figure 2, crossed section reflects the percent ratio in number of accesses to indirect blocks required to map a user buffer with several grouping factors. Even when only the first indirection level is shown in the figure, the number of accesses avoided is almost the 80% for a FBB factor of 5 blocks. To improve accesses smaller than the former threshold, 5 blocks are always premapped.

Minimizing indirect block accesses highly reduces the number of operations required to execute FBB mapping and the number of operations requested to the ParServer, which increases scalability. The improvement of the mapping time when using the *ParFiSys* FBB correspondence compared with a traditional FBB correspondence is shown in figure 2. This figure shows the experimental results obtained by reading a 10 MBytes file, using 8 Kbyte blocks, with request size varying from 8 KBytes to 512 KBytes. The FBB factor is equal to the request size. As can be seen, the improvement

in time is almost exponential for *ParFiSys*, which is highly affected by the request size. For accesses larger than 128 KBytes, the improvement is very small, so it can be considered as a *maximum FBB factor*. A similar test to evaluate write operations shown an analogous behavior.

FBION, FBC, and FBD depends on the distribution strategy defined by the user or the file system to map data on I/O devices. The FBION correspondence must determine the relation between a file block $x$ and the node where it is located. Establishing the former correspondences for extended and cyclic partitions having an non uniform layout of controllers and devices is heavy. To enhance the former computations, some information about the distributed partition and file system is computed when the file system is created and stored in the superblock. A *symmetrical* layout which greatly improves the mapping functions, as showm below for a cyclic partition with $c$ controllers, $d$ devices , and $s$ blocks per device, are:

$$FBION = \frac{x \bmod pd}{cd} \tag{2}$$

$$FBC = \frac{(x \bmod pd) \bmod cd}{d} \tag{3}$$

$$FBD = ((x \bmod pd) \bmod cd) \bmod d \tag{4}$$

where $pd = n * c * d$ is the number of devices in the partition, and $cd = c * d$ is the number of devices per ION.

## 4 Conclusion

The concurrent architecture of *ParFiSys* provides high concurrency and parallelism in ParClients and ParServers, reducing the latency of services and increasing the overall performance. Multi-level mapping is a very flexible and scalable mechanism to connect the user data image with the file system internal image. The correspondence used to map user space to file blocks provides a very good performance because of grouped algorithms and parallelization. Data distribution algorithms have shown a high flexibility to map data onto different kind of file systems, becuase of the generality of the partition structure. Finally, it is very important to remark that users may define data mapping functions to be used by *ParFiSys*, even when three mapping strategies are predefined internally.

## References

1. R. Bordawekar, J. Rosario, and A. Choudhary. Design and Evaluation of Primitives for Parallel I/O. In *Supercomputing 93*, pages 452–462. IEEE, 1993.
2. J. Carretero, F. Pérez, P. De Miguel, F. García, and L. Alonso. ParFiSys: A Parallel File System for MPP. *ACM SIGOPS*, 30(2):74–80, April 1996.
3. P. Corbett, S. Johnson, and D. Feitelson. Overview of the Vesta Parallel File System. *ACM Computer Architecture News*, 21(5):7–15, December 1993.
4. E. DeBenedictis and J. M. del Rosario. nCUBE Parallel I/O Software. In *Eleventh Annual IEEE International Phoenix Conference on Computers and Communications (IPCCC)*, pages 117–124, April 1992.