# MAPFS-Grid: A Flexible Architecture for Data-Intensive Grid Applications

María S. Pérez[1], Jesús Carretero[2], Félix García[2], José M. Peña[1], and Víctor Robles[1]

[1] DATSI, FI, Universidad Politécnica de Madrid, Spain
[2] Departamento de Informática, Universidad Carlos III de Madrid, Spain

**Abstract.** Grid computing constitutes one of the most important computing paradigms appeared in the last decade. Data grids are grid system, whose main concern is the efficient and reliable management of data. These systems have had a growing interest due to the increasing number of applications using a huge amount of data, known as data-intensive applications. They present problems related to both grid systems and I/O systems. This paper describes MAPFS-Grid, a flexible and high-performance platform for data grid applications.

## 1 Introduction

I/O system constitutes a bottleneck in most of the current computing systems, due to its poor performance. The usage of parallel file systems is one of the most widely used alternative for avoiding this problem, traditionally known as *I/O crisis*.

Because of the improvements in hardware and, moreover, the decreasing prices of computer components, clusters have become an appropriate alternative in parallel and distributed computing. Nevertheless, the high demand of both computation and storage, needed by a great number of applications, which manage a huge amount of data, requires the usage of new technologies, such as grid computing.

Grid computing becomes one of the most important computing paradigms appeared in the last decade. This kind of computing allows applications to use low-load periods of all the nodes connected to a high-speed network. Unlike a conventional cluster, a high-performance grid infrastructure involves a heterogenous set of computing nodes, situated in different locations, and using different structures and policies. Grid technology allows hard-computing problems demanding huge storage facilities to be solved in an efficient way. In fact, one of the major goals of grid computing is to provide an efficient access to data, being data-intensive grid applications (or data grids, in short) one of the most relevant grid architectures [6]. Data-intensive applications usually make use of data management systems, by means of data mining and data warehousing techniques or other information management algorithms, which require efficient information retrieval capacities and a global and broad storage space. A data grid can fulfill the needs of these environments.

This paper presents MAPFS-Grid, a multiagent architecture, based on MAPFS [12], whose main goal is the deployment of MAPFS capabilities in a grid environment.

The outline of this paper is as follows. Section 2 describes the role of I/O access in grid computing. Section 3 presents MAPFS-Grid as a flexible infrastructure for data-intensive grid applications. Section 4 describes MAPFS-Grid architecture. Section 5 describes how MAPFS-Grid can interact with other data grid infrastructures. Section 6 shows the results obtained for the evaluation of applications using MAPFS in a grid environment. Section 7 shows the related work. Finally, section 8 summarizes our conclusions and suggests further future work.

## 2 The Role of I/O Access in Grid Computing

Nowadays, there are a huge number of applications creating and operating on large amounts of data, e.g. data mining systems extracting knowledge from large volumes of data. Existing data-intensive applications have been used in several domains, such as physics, climate modeling, biology or visualization.

As we mentioned previously, data-intensive grid applications try to tackle the problems originated by the needs of a performance-full I/O system in a grid infrastructure. In these architectures, data sources are distributed among different nodes. Also, a typical data grid requires access to terabyte or higher sizes datasets. For example, high-energy physics may generate terabytes of data in a single experiment. Accesses to data repositories must be made in an efficient way, in order to increase the performance of the applications used in the grid. Furthermore, data-intensive grid applications have several functional requirements and access patterns.

Currently, there are different systems that offer services to access resources in a data grid. Accessing heterogeneous resources with interfaces and different functionalities is solved, in most of the cases, by means of new services that offer a uniform access to different types of systems. Examples of this kind of systems are Storage Resource Broker (SRB) [1], DataCutter [3], DPSS [16], and BLUNT [9]. All these systems use replication to improve the I/O performance and reliability.

In any case, the I/O system must be flexible enough to match data-intensive applications demands. The usage of hints, caching and prefetching policies or different data distribution configurations can reduce latency and increase I/O operations performance.

## 3 MAPFS-Grid Features

As we mentioned previously, a key feature of data grids infrastructures is the flexibility. MAPFS is a multiagent architecture, which provides this property mainly by means of three approaches:

1. System topology configuration: Ability to change system topology, setting the I/O nodes and their relationships. This feature is achieved by means of *storage groups*.
2. Access pattern specification: Although MAPFS is a general purpose I/O system, it can be configured in order to adapt to different I/O access patterns. The main configuration parameters of the MAPFS system are: (i) I/O caching and prefetching, approaches that increases the I/O operations efficiency, because of the optimal usage of disk caches, and (ii) usage of hints on future access patterns. MAPFS offers an independent API, different from the I/O operations API, which allows applications to configure the access patterns, which are translated to hints by the I/O system. All these features can be configured through the usage of *control user operations*.

3. There are different reasons to allow some functionalities (such as caching or prefetching) to run in parallel on different nodes and even in data servers. Moving executions to data servers may reduce network latency and traffic. The *agent* technology is a suitable framework for integrating these functions in the storage system, because of its adaptability to different domains and the agents autonomy.

MAPFS-Grid takes advantage of all these features with the aim of building a flexible and powerful infrastructure for data grids.

## 3.1 MAPFS-Grid Storage Groups

A storage group is defined in MAPFS as a set of servers clustered as groups. These groups take the role of data repositories. These groups can be built applying several policies, trying to optimize accesses to storage groups. Some significant policies are:

– Grouping by server proximity: Storage groups are built based on the physical distribution of data servers. Storage groups are composed of servers in close proximity to each other. This policy optimizes the queries addressed to a storage group because of the similar latency of messages sent to servers.
– Grouping by server similarity: Storage groups are composed of servers with similar processing capacity. This policy classifies storage groups in different categories, depending on their computational and I/O power.

The system topology can be changed dynamically. In this case, data must be reconstructed, degrading the performance of the I/O system. In order to avoid data reconstruction, MAPFS defines two types of storage groups, main storage groups and secondary groups, which form a lattice structure between them [14]. This approach postpones data reconstruction until the system schedules a defragmentation operation, which is used for deleting secondary groups and simplifying the storage system description.

## 3.2 Applications Access Pattern Specifications in MAPFS-Grid

Hints are structures known and built by the I/O system, which are used for improving the read and write operations performance. MAPFS uses these hints to access data. For example, storage systems using hints may provide greater performance since they use this information to decrease cache faults and to prefetch data most probably used in next executions. In other words, the more information it has been used, the less uncertainty in the future access guesses and, therefore, the better prefetching and caching results. In MAPFS, hints can be obtained in two ways: (i) given by the user, that is, the user application provides the necessary specifications to I/O system, and (ii) built by the multiagent subsystem. If this option is selected, the multiagent system must analyze the access pattern of the applications in order to build hints for improving data access. This feature can be achieved using statistical methods or historical logs.

If hints are provided by the user application, it is necessary for the system to provide syntactic rules for setting the system parameters, which configure the I/O system. On the other hand, if the multiagent subsystem creates the hints, it is also necessary to store them in a predefined way. In any case, lexical and syntactic rules must be introduced in the system.

The system is configured through several operations, which are independent of the I/O operations, although these last ones use the former operations. The configuration operations are divided into: (i) Hints Setting Operations, operations for establishing the hints of the system (they can set and get the values of the different fields of the hints), and (ii) Control User Operations, higher level operations that can be used directly by the user applications to manage system performance.

We refer the reader to [13] for a more detailed description of this interface.

### 3.3 MAPFS-Grid Agents Hierarchy

MAPFS uses an agent hierarchy, which solves the information retrieval problem in a transparent and efficient way. The taxonomy of agents used in MAPFS is composed of:

– Extractor agents: They are responsible for information retrieval, invoking parallel I/O operations.
– Distributor agents: They distribute the workload to the extractor agents. These agents are placed at the higher level of the agents hierarchy.
– Caching and prefetching agents: They are associated with one or more extractor agents, caching or prefetching their data.
– Hints agents: They must study applications access patterns to build hints for improving data access.

## 4 MAPFS-Grid Architecture

MAPFS file system uses as underlying hardware infrastructure a cluster of workstations [12]. Clusters are, in some sense, the predecessors of the grid technology.

Supercomputers have been replaced by clusters of workstations in a huge number of research projects. A relevant sample of this fact is the evolution of particle physics computation in CERN (European Organisation for Nuclear Research). Experiments over its previous LEP accelerator were made in IBM and CRAY supercomputers. In the early 1990's, the experimentation environment was replaced by tens of RISC processors. In the late 1900's, clusters of hundreds of Linux PCs were used in the accelerator. The new LHC accelerator will be used by 2007. This accelerator demands new solutions, since it has to manage several petabytes of data. 200,000 interconnected nodes are said to be necessary for the LHC accelerator, which arises both financial and technical difficulties. The solution involves resource sharing of a huge number of geographical distributed institutions, by means of grid computing. Analogously, MAPFS has modified its architecture aiming at the achievement of the grid computing advantages.

MAPFS is based on a client-server architecture using general purpose servers, providing all the MAPFS management tasks as specialized clients. In the first prototype, we use NFS servers. NFS has been ported to different operating systems and machine platforms and is widely used by many servers worldwide. Therefore, it is very easy to add new data repositories to the data grid infrastructure. The only requirement of these data servers is to use NFS and export a directory tree to data grid users. Data is distributed through the servers belonging to a storage group, using a stripe unit.

On the client-side, it is necessary to install MAPFS client, which provides a parallel I/O interface to the servers. This module is implemented with MPI (Message Passing Interface)[10], the standard message passing interface, widely used in parallel computing. MPI allows different processes to run in parallel over MAPFS. Nevertheless, this technique is not suitable for the dynamic interconnection of heterogeneous nodes, because MPI is a static solution, which must know a priori the IP address of all the nodes of the topology. Thus, MAPFS-Grid requires grid technology, interoperable with MPI, because MAPFS is based on this interface. In order to fulfill these requirements, MAPFS-Grid uses MPICH-G2 [4], such as described in next section.

## 5 Interoperability with Other Grid Architectures

One of the major goals of a data grid infrastructure is to provide access to a huge number of heterogeneous data sources. In this sense, it is important that MAPFS-Grid can interoperate with other grid architectures, giving access to their data repositories. Because MAPFS is implemented with MPI, its integration with other grid infrastructures is relatively simple, using MPICH-G2, a grid-enabled implementation of the MPI, which makes possible running MPI programs across multiple computers at different sites.

Applications can use MAPFS-Grid together with other grid architectures. In this case, it is possible to extend storage groups with other nodes accessible through the Globus services. Concretely, the Global Access Secondary Storage (GASS) service [2] is used in order to stage programs to remote machines and to support efficient communication in dynamic grid environments. The integration between MAPFS and GASS is not redundant, because GASS does not provide the full functionality of a parallel file system. MAPFS provides a rich parallel interface, which can be used in wide area computing with the aid of GASS and other Globus services.

## 6 MAPFS-Grid Evaluation

In our implementation, we need to evaluate the performance of: (i) storage groups, (ii) control user operations, and (iii) multiagent subsystem.

In order to measure the performance of the first aspect, experiments were run in two different storage groups, which use the server similarity grouping policy, because of the technical differences of both groups. The first storage group ($G_1$) is composed of four nodes Athlon 650MHz, with 256 MB of RAM memory, connected by a Gigabit network. The second storage group ($G_2$) has six nodes Intel Xeon 2.40GHz, with 1GB of RAM memory with a Gigabit network. The storage group $G_2$ provides better performance. However, the storage group $G_1$ offers bigger storage capacity. These two storage groups constitutes a possible topology of our data grid. Our experiment consists in a process per node running a multiplication of two matrices, where the matrices are stored in the grid, using MAPFS-Grid as underlying platform. The resultant matrix is also stored in a distributed fashion. A prefetching multiagent subsystem is used, which is responsible for prefetching rows and columns of the matrices. In this case, hints provided by the applications are the indexes of the matrix row and the matrix column of the element calculated in every iteration. It is possible to prefetch data to be used later in the executions, using this information. The multiagent subsystem obtains optimum values for the prefetching phase. In this way, we evaluate the usage of control user operations and the performance of the multiagent subsystem.

This experiment was compared to another one, which consists in multiplying the same matrices stored in the local disk through the usage of a traditional I/O system. The size of the matrices is 100 MB.

Figure 1 shows the speedup of the MAPFS solution for the group $G_1$ versus local solution, varying the access size used in the I/O operations. As can be seen, the speedup is very close to the maximum speedup. Figure 2 represents the execution time of the groups $G_1$ y $G_2$ for the matrix multiplication. As we previously mentioned, the storage group $G_2$ provides better results, although the storage group $G_1$ provides higher storage capacity. The usage of different policies in MAPFS-Grid allows applications to take advantage of the flexibility of this infrastructure, depending on their current needs.

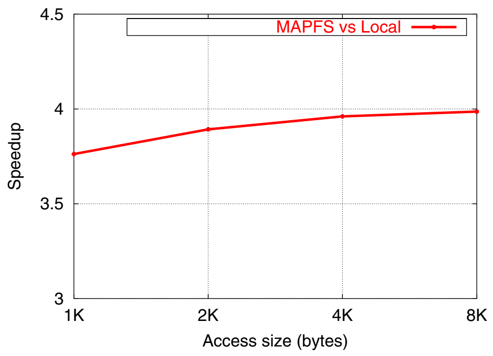

**Fig. 1.** Speedup of the MAPFS-Grid solution (group $G_1$) versus Local solution.

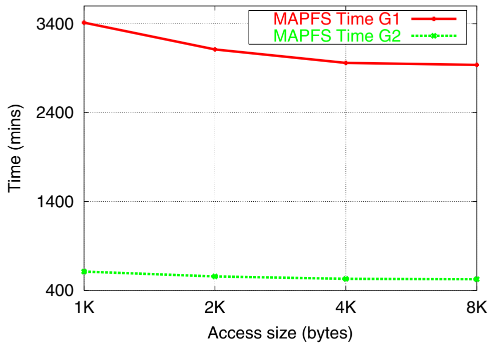

**Fig. 2.** Comparison between different storage groups.

## 7 Related Work

Grid technology provides a framework in which heterogeneous and distributed computing resources can be shared among several organizations and institutions, through high-speed networks, with the aim of solving high-cost and data-intensive problems. Foster et al. [5] compares grid technology to electrical power grid, in the sense that a user must be able to use computational resources in the same way than electrical power, that is, everywhere, with a reliable service, and having an acceptable cost.

Distributed scientific applications running in a high-speed network of 17 USA research centers were shown in SuperComputing'95 congress. This demonstration constituted the starting poing of several research projects related to the distributed resource-sharing. One of the first grid projects was the NASA Information Power Grid (IPG), which allows NASA resources to be integrated and managed.

Grid owns a *de facto* standard, knows as Globus. Globus [7] provides a software infrastructure, which includes basic protocols and services for grid applications.

Additionally, several researchers and commercial companies are investigating topics related to data-intensive grids and I/O systems used in computational grids. The problems tackled by these research lines are similar to those discussed in this paper. Armada [11] is a framework that allows grid applications to access datasets distributed across a computational grid. Applications combine modules called *ships* into graphs called *armadas*.

The Remote I/O (RIO) library [8] is a tool for accessing to files located on remote file systems. RIO follows the MPI-IO interface. MAPFS also provides this feature.

Kangaroo [15] belongs to the Condor grid project and it is a reliable data movement system that keep applications running. Kangaroo service continues to perform I/O operations even if the process that initiated these requests fails.

Legion is an object-oriented infrastructure used in distributed computing. LegionFS [17] provides UNIX-style I/O operations, using Legion services such as naming or security. Unlike LegionFS, MAPFS provides a rich parallel I/O interface.

MAPFS-Grid allows applications to access remote data in an efficient way, making possible access to storage groups belonging to other data grids, by means of the usage of MPICH-G. Furthermore, data access patterns configuration provides flexibility to applications using MAPFS-Grid. This last characteristic is different from the rest of the systems previously described.

## 8 Conclusions and Future Work

In this work we have presented MAPFS-Grid as an extension of the MAPFS I/O architecture for data-intensive grid applications. MAPFS-Grid constitutes a new data grid infrastructure, which provides flexibility and dynamic reconfiguration to applications mainly by means of three approaches: (i) storage groups, (ii) access patterns specifications, and (ii) a multiagent subsystem responsible of running specific functionalities. These features have been evaluated through the implementation of a scientific application, achieving a speedup very close to the maximum one.

As future work, it would be interesting to evaluate the performance of the system with other topologies and other kind of applications.

## References

1. C. Baru, R. Moore, A. Rajasekar, and M. Wan. The SDSC storage resource broker. In *Proceedings of CASCON'98*, Toronto, Canada, 1998.
2. Joseph Bester, Ian Foster, Carl Kesselman, Jean Tedesco, and Steven Tuecke. GASS: A data movement and access service for wide area computing systems. In *Proceedings of the Sixth Workshop on Input/Output in Parallel and Distributed Systems*. ACM Press, 1999.
3. M. D. Beynon, R. Ferreira, T. Kurc, A. Sussman, and J. Saltz. DataCutter: Middleware for filtering very large scientific datasets of archival storage systems. In *Proceedings of the 2000 Mass Storage Systems Conference*. IEEE Computer Society Press, Mar 2000.
4. I. Foster and N. Karonis. A grid-enabled MPI: Message passing in heterogeneous distributed computing systems. In *Proceedings of SC'98*. ACM Press, 1998.
5. I. Foster and C. Kesselman, editors. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 1999.
6. Ian Foster. *Computational Grids*. 1999. Chapter belonging to [5].
7. Ian Foster and Carl Kesselman. Globus: A metacomputing infrastructure toolkit. *The International Journal of Supercomputer Applications and High Performance Computing*, 11(2):115–128, Summer 1997.
8. Ian T. Foster, David Kohr, Rakesh Krishnaiyer, and Jace Mogill. Remote I/O fast access to distant storage. In *Proceedings of the IOPADS 1997*, pages 14–25, 1997.
9. M.R. Martinez and N. Roussopoulos. MOCHA: A self-extensible database middleware system for distributed data sources. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Dallas, TX, May 2000.
10. The Message Passing Interface (MPI) standard. *http://www-unix.mcs.anl.gov/mpi*
11. Ron Oldfield and David Kotz. Armada: a parallel I/O framework for computational grids. *Future Generation Computer Systems*, 18(4):501–523, 2002.
12. María S. Pérez, Jesús Carretero, Félix García, José M. Peña, and Víctor Robles. A flexible multiagent parallel file system for clusters. *International Workshop on Parallel I/O Management Techniques (PIOMT'2003) (Lecture Notes in Computer Science)*, June 2003.
13. María S. Pérez, Ramón A. Pons, Félix García, Jesús Carretero, and Víctor Robles. A proposal for I/O access profiles in parallel data mining algorithms. In *3rd ACIS International Conference on SNPD*, June 2002.
14. María S. Pérez, Alberto Sánchez, José M. Peña, Víctor Robles, Jesús Carretero, and Félix García. Storage groups: A new approach for providing dynamic reconfiguration in databased clusters. In *2004 IASTED Conference on PDCN (To appear)*, February 2004.
15. Douglas Thain, Jim Basney, Se-Chang Son, and Miron Livny. The Kangaroo approach to data movement on the grid. In *Proceedings of the Tenth IEEE Symposium on High Performance Distributed Computing (HPDC10)*, August 2001.
16. B. Tierney, J. Lee, W. Johnston, B. Crowley, and M. Holding. A network-aware distributed storage cache for data-intensive environments. In *Proceedings of the Eighth IEEE International Symposium on High Performance Distributed Computing*, pages 185–193, Redondo Beach, CA, Aug 1999.
17. Brian S. White, Michael Walker, Marty Humphrey, and Andrew S. Grimshaw. LegionFS: A secure and scalable file system supporting cross-domain high-performance applications. In *Proceedings of the IEEE/ACM Supercomputing Conference (SC2001)*, November 2001.