A High Performance Suite of Data Services for Grids

Alberto Sánchez [a], María S. Pérez [b], Jesús Montes [b], Toni Cortes [c]

[a] *Universidad Rey Juan Carlos. Spain. alberto.sanchez@urjc.es*

[b] *DATSI. FI. Universidad Politécnica de Madrid. Spain. {mperez,jmontes}@fi.upm.es*

[c] *Universidad Politécnica de Cataluña and the Barcelona Supercomputing Center. Spain. toni.cortes@bsc.es*

Abstract

The last advances in computation have made feasible the resolution of very complex problems. Grid computing has matured to currently become one of the most successful initiatives for dealing with these challenging problems, which usually require tackling huge volumes of data. Although the main goal of the grid technology is sharing resources among several virtual organizations, few research works are oriented to increase the performance of grid solutions.

MAPFS-Grid is a complete suite of components for providing high performance access to huge volumes of data in a grid environment. MAPFS-Grid is composed of three different services: (i) Parallel Data Access Service, a WSRF-compliant grid service; (ii) MAPFS-DSI, a GridFTP-compliant service, and (iii) MAPFS-DAI, an OGSA-DAI-compliant service. Each one is suitable for different scenarios, which will be described along the article.

This paper describes and evaluates the performance of MAPFS-Grid, showing the main advantages of MAPFS-Grid vs other proposals.

Key words: Data access services, data exchange, data-intensive applications, data grids, data storage.

1 Introduction

Advances in the computational field have made possible to develop increasingly complex applications to face new and challenging problems. Most of these applications require the management and analysis of large volumes of data, which range from terabytes to petabytes. These data-intensive applications can be found in several domains, including Physics [27], climate modeling [34], Biology [55] or visualization [23]. The I/O phase usually constitutes the bottleneck of these applications.

Data grid [15,6,28] aims at developing suitable solutions to these kinds of applications by means of grid-based tools. Indeed, a data grid is specifically designed to store, manage, and provide reliable access to data.

Data grid services must provide a series of features in order to achieve maximum performance:

- Ability to search through numerous available datasets.
- Ability to select suitable computational resources to perform data analysis.
- Ability to manage access permissions.
- Intelligent resource allocation and scheduling.

Due to the basic grid principles, the environment is characterized by its heterogeneity. In the case of a data grid, this includes different storage systems, data access mechanisms, data access policies, and data formats. The data grid management infrastructure must act as an abstraction layer that provides a common, standard and efficient procedure to access the information stored.

Although data grid allows heterogeneous data resources to be shared, only few research works in the field of data grid are oriented to increase the performance of these solutions [32,9].

The aim of this work is to develop a complete suite of services for high performance access to huge volumes of data in a grid environment. Our approach, named MAPFS-Grid [40], is intended to provide a high performance access by means of the following kind of services:

(1) A generic WSRF-compliant data access service, which uses Simple Object Access Protocol (SOAP) and Web services technology. This proposal follows the OGSA guidelines [22], which propose Web services as basic technology for building grids.
(2) A performance-oriented data access service based on GridFTP and built within MAPFS-Grid. GridFTP [2] is a high-performance and reliable file transfer protocol, although it does not follow strictly the OGSA scheme.
(3) An OGSA-DAI-compliant service for providing a uniform access and better performance than OGSA-DAI [4]. OGSA-DAI provides a uniform way of querying, accessing, updating, and transforming different type of data resources by means of web services.

All these three scenarios cover the needs of most data grid-based applications. Moreover, services provided by MAPFS-Grid are incorporated within the generic architecture of a grid. Figure 1 shows the scheme in which MAPFS-Grid is integrated. MAPFS-Grid uses other core grid services and supports data management

and storage system, allowing other services (such as the replica manager) to offer more complex functionalities. All these other services are out of the scope of this paper, although are integrated together with MAPFS-Grid.

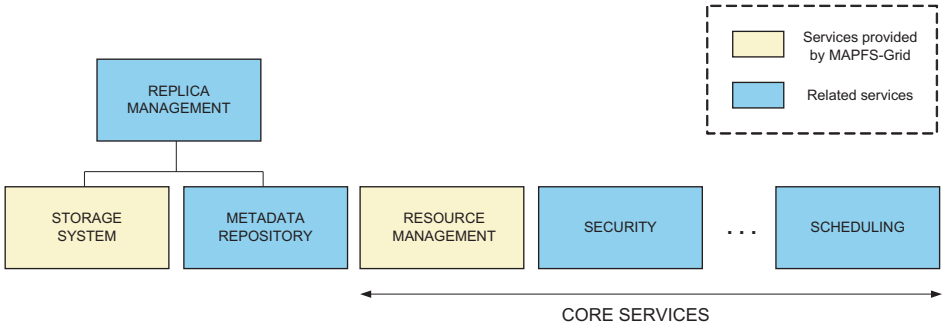


Fig. 1. Services provided by MAPFS-Grid and related

The outline of this paper is as follows. Section 2 describes our proposal, MAPFS-Grid, as a suite of three different data services intended to provide high performance in different scenarios. Section 3 shows different experimental results, which evaluate the main characteristics of MAPFS-Grid. Section 4 enumerates different works related to our proposal. Finally, in Section 5 the conclusions of our work and the open research lines are described.

2 MAPFS-Grid, a suite of services for accessing large volumes of data

As we have seen in the previous section, different needs arise in data management and access in grids. We have noticed three important aspects related to these needs:

(1) An efficient data service that manages file resources following the OGSA architecture is required. The Web services technology is suitable for managing services and resources through Web Service Resource Framework (WSRF) [37]. As far as we know, there are not WSRF-based data services designed to increase the performance of the I/O operations.
(2) The most important drawback of the previous scenario is the low performance exhibited by web services. In fact, the use of XML and SOAP as transfer protocol is not appropriate for performance-critical applications [16]. Although there are different proposals for dealing with this decrease of performance [52], [45], [33], none of them are suitable for scenarios demanding high throughput. In this context, GridFTP is an optimized protocol for transfer of large files, since it is not based on SOAP transfer. Additionally, GridFTP extends the basic FTP protocol to support data transfer among multiple servers (striping). Furthermore, GridFTP enables the use of multiple TCP streams in parallel from a source to a sink (parallelism) [3].

(3) Every grid project usually provides "ad hoc" solutions to the data management. A data service often offers a *native* interface, which does not provide interoperability with other I/O systems. OGSA-DAI [4] has emerged to provide a uniform access to data sources in a grid environment. However, OGSA-DAI is not focused on the performance of the I/O operations. Therefore, providing a bridge between the interoperability and the performance optimization is an important need of current data grid projects.

MAPFS-Grid tries to build a generic framework where all these problems can be solved, by means of the definition of different services, suitable for these three identified scenarios.

MAPFS-Grid makes use of MAPFS [39], a high-performance parallel file system for clusters of workstations. MAPFS (*Multi Agent Parallel File System*) has been developed in the Universidad Politécnica de Madrid since 2003. The main contribution of MAPFS is the conceptual use of agents to provide applications with new properties, with the aim of increasing their adaptation to dynamic and complex environments. MAPFS is based on a multiagent architecture that offers features such as data acquisition, caching, prefetching, and the use of hints.

The feasibility of the combination between MAPFS and MAPFS-Grid is due to the fact that grid environments are composed of different and heterogeneous resources, being clusters one of the most used because of its good relation power vs. cost. Thus, it is possible to improve the grid data operations through parallel accesses into the clusters resources. MAPFS distributes data stripes over all the nodes of a cluster. On the other hand, MAPFS-Grid allows heterogeneous servers connected by means of a wide-area network to be used as data repositories, by storing data in a parallel way through all the clusters and individual nodes which compose the grid.

The heterogeneity of grid environments makes the application of parallelism difficult. In fact, since every resource of the grid can be composed of several components (e.g., clusters of workstations), it is necessary to optimize the I/O performance of every resource before tackling the global I/O optimization. Therefore, MAPFS-Grid, as a complete suite of services, provides two levels of software parallelism in a grid:

(1) The high level provides parallelism among the grid storage elements, that is, *inter-storage element parallelism*.
(2) The low level provides parallelism among the set of nodes of each cluster, that is, *intra-cluster parallelism*. At this level, MAPFS is applied.

Both levels are integrated and cooperate with the aim of providing an enhanced I/O bandwidth. Figure 2 shows the double parallelism of MAPFS-Grid. The inner

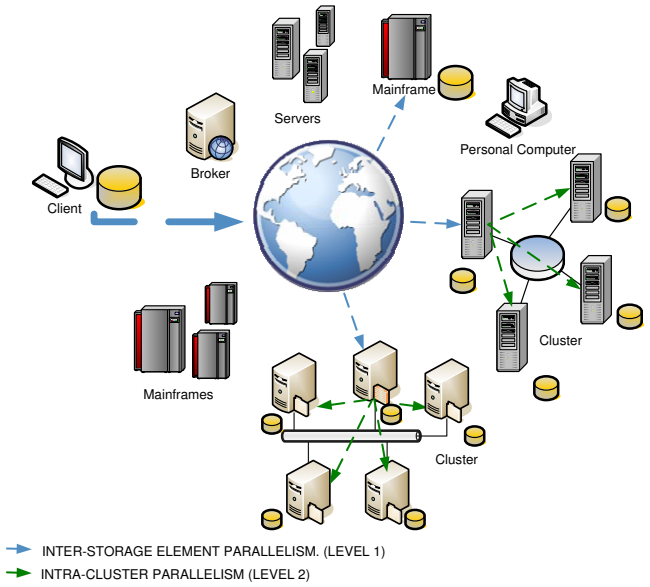


Fig. 2. Two levels of parallelism in MAPFS-Grid

level is achieved only if the storage element is a cluster of nodes. The outer level is provided among a set of storage elements, at grid level. Next subsections describe the three MAPFS-Grid services.

2.1 MAPFS-Grid Parallel Data Access Service (PDAS)

Our first proposal is to provide a grid-like interface to MAPFS. This WSRF-compliant service, named PDAS, allows parallel I/O operations to be made in a cluster environment. The conception of this service comes from Data Access and Integration Service (DAIS) [17]. PDAS is an adaptation of this concept from the performance and parallelism viewpoints.

The two levels of parallelism provided by PDAS are shown in Figure 3. The level 1 parallelism is provided by several PDAS (in every storage element), which give support to a distributed data repository. The level 2 parallelism is offered directly by MAPFS, in those storage elements which are clusters. As this figure shows, data to be transfered are divided in blocks which are sent to each storage element. These data blocks are internally divided and sent to each node if the storage element is a cluster.

The main advantage of PDAS is that constitutes a WSRF-compliant grid service, which provides reasonably good performance and it is easy to deploy in a grid scenario where the main components are clusters of workstations.

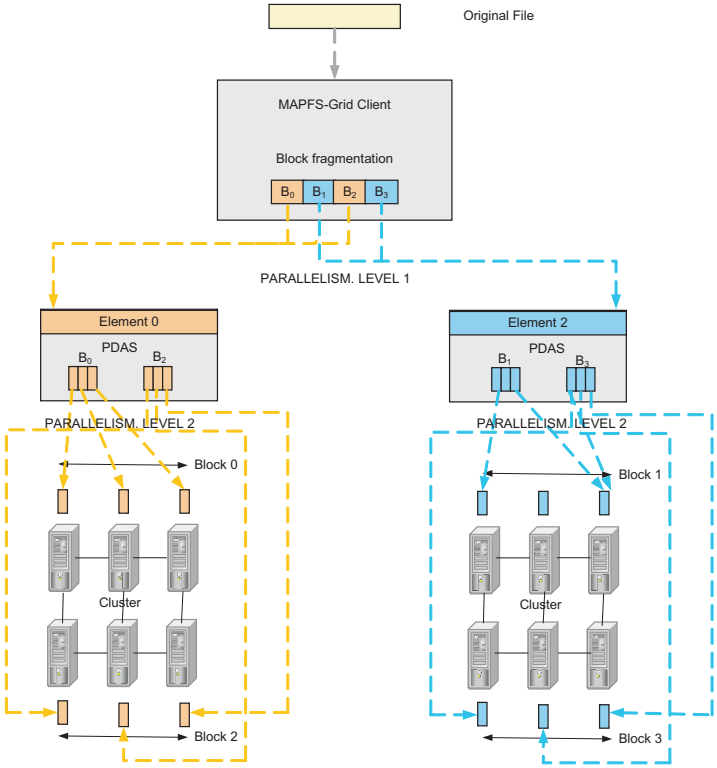


Fig. 3. PDAS providing two levels of parallelism

2.2 MAPFS Data Storage Interface (MAPFS-DSI)

As previously mentioned, one of the limitations of grid services, and thus MAPFS-Grid PDAS, is the use of SOAP, which introduces a noticeable amount of overhead. In those services demanding very high performance, it is necessary to find alternatives to SOAP. GridFTP is one of them, since it provides several features for optimizing the data transfer from a source to a sink, following a client-server scheme.

GridFTP provides two different approaches for increasing the performance of the transfer between client and servers, that is, parallelism and striping. The parallelism consists in using multiple TCP streams in parallel from a source to a sink. On the other hand, the striping characteristic allows GridFTP to transfer data among multiple servers.

Focusing on the GridFTP server, it is possible to optimize its performance by modifying one of its modules. This module is the Data Storage Interface (DSI), whose responsibility is to read and write to the local storage system. The DSI is composed of several function signatures, which must be filled with suitable semantics to provide a specific functionality. An important characteristic is the fact that the DSI can be loaded at runtime. We have used this flexibility of the GridFTP server for trans-

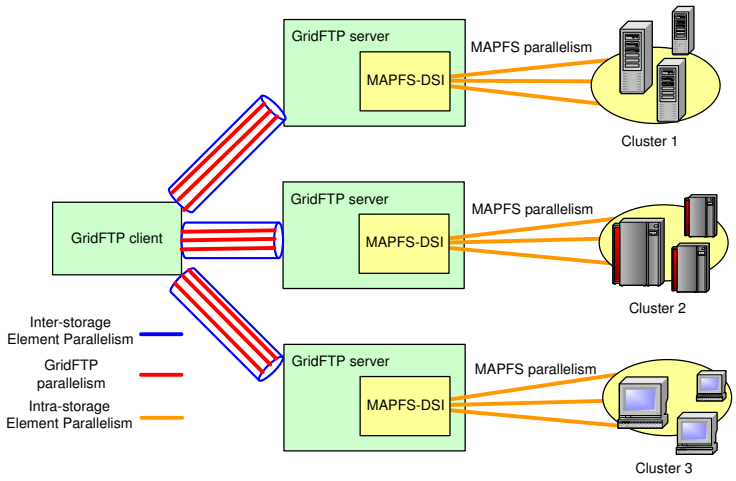


Fig. 4. MAPFS-DSI within a data transfer scenario

forming the I/O operations. MAPFS I/O routines are used instead for enhancing the server [1] . The result is MAPFS-DSI. MAPFS-DSI enables GridFTP clients to read and write data in a storage system based on MAPFS. As the architecture for MAPFS is a cluster of workstations, the GridFTP server should be the master node from a cluster of workstation, where MAPFS is installed.

In order to provide all these characteristics, MAPFS-DSI is composed of two modules:

(1) The DSI module, which acts as interface between the GridFTP server and the I/O system. However, the I/O operation is not directly performed by this module. The second module is responsible for this task.
(2) The MAPFS driver, which performs the actual I/O operation. This driver invokes the specific MAPFS operations.

MAPFS-DSI is embedded within the general scenario in which GridFTP is used. As we can see in the Figure 4, there are two independent parts of the architecture that can improve the performance of a data transfer operation. Firstly, the specific features of GridFTP (TCP stream parallelism and striping), which can be used in any GridFTP server. Secondly, the parallel access provided by MAPFS. This implies that the use of MAPFS within the GridFTP server offers two levels of data parallelism, avoiding that the server storage system becomes a bottleneck in the whole data transfer process. MAPFS-DSI offers great flexibility, since several combinations of both levels of parallelism can be used in different configurations.

[1] A complete list of these functions can be found in [41].

2.3 MAPFS Data Access and Integration (MAPFS-DAI)

Due to the existence of an extensive number of data sources and storage systems in different grid projects, the interoperability among them is playing a revelant role in the grid research.

OGSA-DAI [31] project intends to provide a uniform access to data resources, being compliant with OGSA [21]. However, the performance of OGSA-DAI is quite poor [2] . MAPFS-DAI constitutes an extension of the OGSA-DAI architecture, whose aim is to increase this performance.

As Figure 5 shows, the MAPFS-DAI architecture is divided into four layers:

(1) Data Layer, composed of data resources. Data resources exposed by MAPFS-DAI are flat and unformatted files. On the other hand, OGSA-DAI gives support to other kind of data resources, such as relational and XML databases.
(2) Business Logic Layer, which is composed of:
   - A suitable data service resource, which is named *File Data Service Resource*, associated to flat and unformatted files.
   - A MAPFS-DAI *accessor*, whose main goal is to control access to the underlying data resource, that is, files. The MAPFS-DAI *accessor* enables *activities* to access data resources. Activities are the operations performed by data service resources. Currently in MAPFS-DAI, we have two activities, one for reading (*FileAccessActivity*) and another one for writing (*FileWritingActivity*), which are compliant with the File Activities defined by OGSA-DAI.
(3) Presentation Layer, which provides the web service interfaces to data services. MAPFS-DAI uses WSRF.
(4) Client Layer, with two components: client application and client toolkit. The client toolkit makes the development of client applications easy by providing useful and simple tools to create the *perform* and *response* documents exchanged between the client and server. Both documents must fulfill the requirements specified by the service schema. In this way, we can optimize the storage system, without changing the client application.

The main disadvantage of providing a uniform access is that the performance is drastically reduced (see Section 3). MAPFS-DAI relieves this decrease thanks to the two levels of parallelism of the MAPFS-Grid philosophy, as the Figure 6 shows:

[2] In [5] it is stated that *"We expect to invest significant effort in engineering good performance..."*

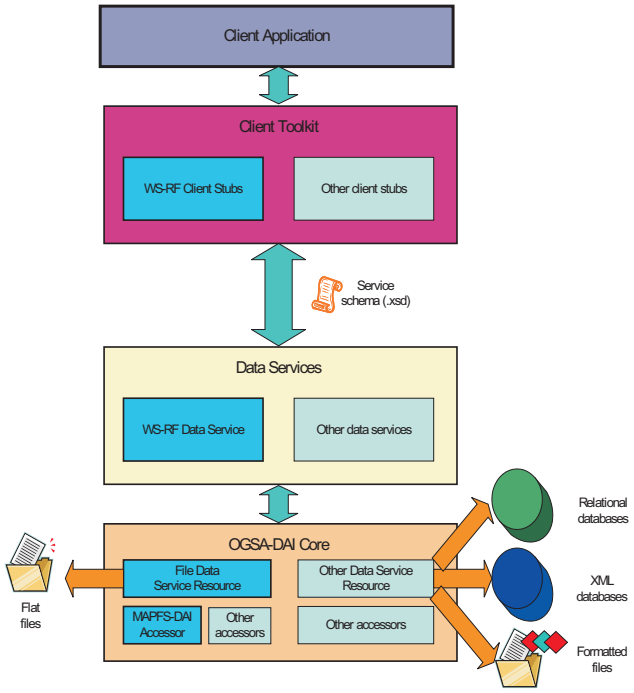


Fig. 5. MAPFS-DAI within the OGSA-DAI Architecture

(1) The high level provides parallelism among the set of storage elements.
(2) The low level provides parallelism among the set of nodes of each cluster.

Therefore, the main advantage of MAPFS-DAI is its interoperability. Every storage element that exhibits the OGSA-DAI interface can be used together with MAPFS-DAI elements. Due to the same interface of OGSA-DAI, several storage systems providing this interface could be accessed in parallel.

As a summary, MAPFS-Grid offers a suite of data services oriented to three paradigmatic scenarios in data grid applications. All these services take advantage of double parallelism.

3 Evaluation

This section analyzes in depth the performance and different benefits of the proposed approach. This analysis aims at demonstrating the efficiency of the proposed two levels of parallelism (intra-cluster and inter-storage elements) vs. the performance obtained by traditional grid data transfer methods.

In this case, the work environment is designed to test the benefits of the proposed grid I/O architecture, obtaining the theoretical and practical limit of the infrastruc-

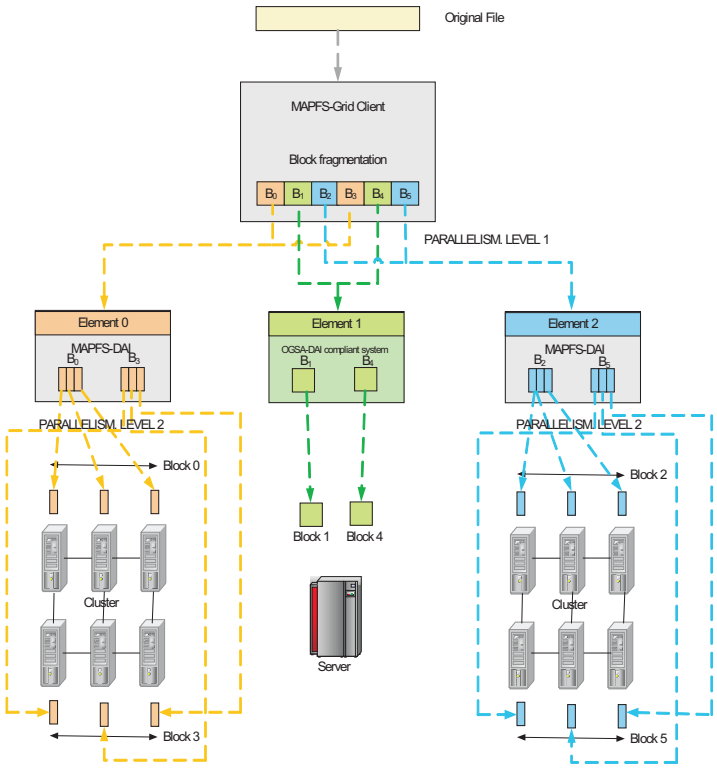


Fig. 6. Two-level parallelism by using MAPFS-DAI and OGSA-DAI compliant storage elements

ture. Thus, it is only constituted by two compound storage elements. *UPM 1* consists of 8 Intel Xeon 2.40GHz nodes with 1 GB of RAM memory, connected by a Gigabit network. The hard disk in each node provides 30 MB/s approximately. *UPM 2* consists of 4 Intel Xeon 3.0GHz nodes with 2 GB of RAM memory, connected by a Gigabit network. The hard disk in each node provides 50 MB/s approximately. Results can be extrapolated to higher grid environments, because as we will see below the technology limits the maximum performance obtained.

To prevent the client from being the throughput bottleneck of this system, a high-performance client is required. An Intel Xeon 3GHz system featuring two hard disks set in a RAID 0 array is selected as client. This client offers a disk read/write bandwidth of around 130 MB/s. Client and storage elements are connected by a Gigabit network.

Experiments have been conducted varying parameters that have a clear influence in the performance of I/O operations on compound storage elements:

(1) Number of nodes that are part of the storage element. Tests are designed to measure the performance improvements obtained by using the whole potential of a cluster or a set of non-individuals nodes working together facing to

traditional single storage elements. The tests analyze the influence of this parameter in the performance. Thus, most of the tests are made over *UPM 1* because it has a higher number of nodes.

(2) Block size BS_{cs} sent between the client c and the storage element s. Since the system is designed to manage massive amounts of data, the block size plays an important role. 64 KB was selected as the most suitable slide size sent by MAPFS parallel to every node. This size corresponds with the maximum data size that DMA drivers can manage. In order to maintain the same conditions among tests, the same block size between client and storage element has to be used. Besides, the chosen block size should take advantage of the parallel access to the whole set of nodes. Therefore, it must be multiple both of the MAPFS slide size (64 KB) and the number of nodes (1,2,4,8).

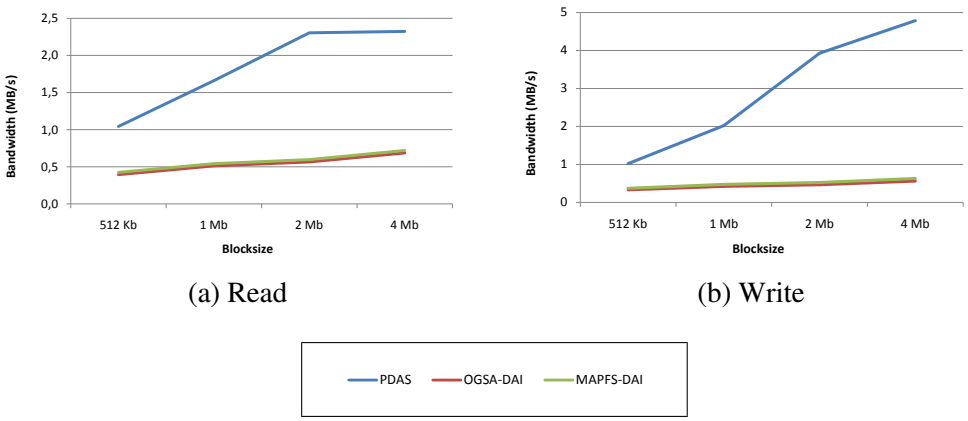
$$BS_{cs} = lcm(1, 2, 4, 8) \times 64KB = 512KB$$

Thus, the system is tested considering the default 512 KB block size, then 1 MB, 2 MB and 4MB block sizes.

(3) File size. Memory hierarchy is designed to take advantage of faster accesses and lower latencies of lower levels, such as cache and memory, vs. the highest levels, like disk storage, to enhance the I/O phase. Memory accesses can be also faster than data sending through network, just as this occurs in the test environment. Therefore, memory hierarchy can hide the benefit obtained by means of the use of parallelism among nodes to access small file size because most of data are accessed in lower levels of memory. Nevertheless, file sizes higher than the memory size cause accessing the highest levels of memory hierarchy to make the I/O operation, where parallel accesses are very beneficial.

Firstly, we have compared the WSRF-based data services, PDAS and MAPFS-DAI, to other kind of systems, like OGSA-DAI. OGSA-DAI assists the access and integration of data, located in separated data sources. Some experiments have been run on *UPM 1* to evaluate the performance of file accesses by means of *intra-cluster parallelism* level provided by PDAS and MAPFS-DAI comparing it to OGSA-DAI architecture. Figure 7 shows the comparison between the performance of read and write operations, respectively, on *UPM 1* by using WSRF-based data services.

The block size is a key factor in the performance of WSRF-based data services. Figure 7 shows that the block size has a clear influence in the performance while the improvement achieved increasing the number of nodes is hidden due to the high influence of the transfer method. In this way, the use of MAPFS-Grid PDAS and MAPFS-DAI only involves a slight improvement according to the parallel use of a higher number of nodes accessed in I/O operations. The improvement is slight in spite of the parallel data access because the high influence data transfer through the network has vs. data access to disk limits the enhancement. Both, MAPFS-Grid PDAS and OGSA-DAI-based solutions are WSRF-based data services, and their performance is limited by the the overhead produced in the transfer protocol used



(a) Read (b) Write

Fig. 7. Performance obtained by the proposed WSRF-based data services to access a 12 GB filesize on *UPM 1*

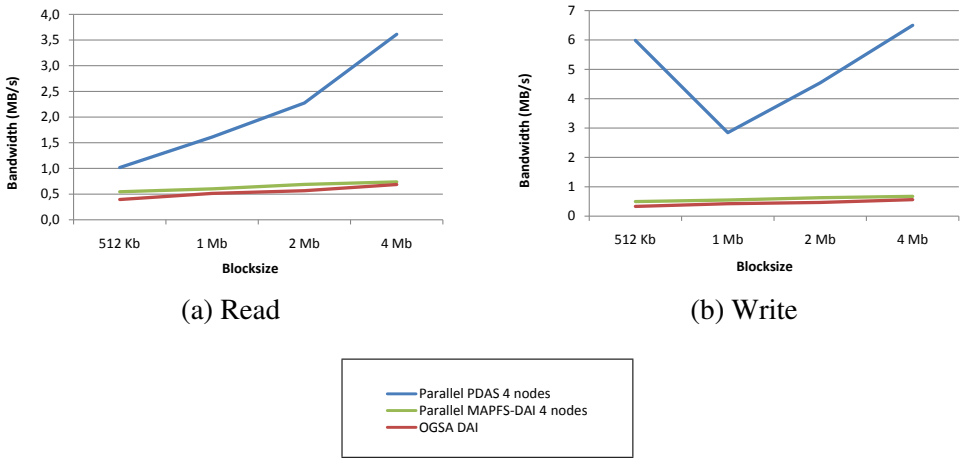
in Web services, SOAP. Thus, a high percentage of time (e.g., 98.7 % on average of the OGSA-DAI operation time) is consumed by performing the file transfer instead of accessing data.

Furthermore, Figure 7 shows that the MAPFS-Grid PDAS access over a single storage element obtains better performance than OGSA-DAI compliant systems. OGSA-DAI has lower performance due to the overhead introduced by the interoperability layer by sending messages and running activities required by the fulfillment of the OGSA-DAI common interface. In this sense, its performance is much lower than MAPFS-Grid PDAS because it is not only a protocol problem, but also a message-passing problem due to the use of the OGSA-DAI's common messages.

As a result, to improve the performance of WSRF-based data services, it is necessary to reduce the transfer time, since the data access time inside the storage element has not a high influence in the results. Parallel techniques applied to the inter-storage element level can be used to decrease the transfer time.

Both elements *UPM 1* and *UPM 2* can be used in parallel by taking advantage of the *inter-storage element parallelism* level, obtaining a performance improvement. In all the figures, we have named *Parallel X* to this approach, where *X* is the specific component (i.e., PDAS, MAPFS-DAI, MAPFS-DSI). Figure 8 shows bandwidths achieved in read and write operations, respectively, on *UPM 1* and *UPM 2* in a parallel way by using WSRF-based data services.

There is a noteworthy improvement in the achieved bandwidth due to the use of the *inter-storage element* level of parallelism. The average improvement when using this level of parallelism is 69.28 % for PDAS and 16.81 % for MAPFS-DAI, obtaining the highest improvements for 512 KB blocks. In the case of MAPFS-DAI,



(a) Read (b) Write

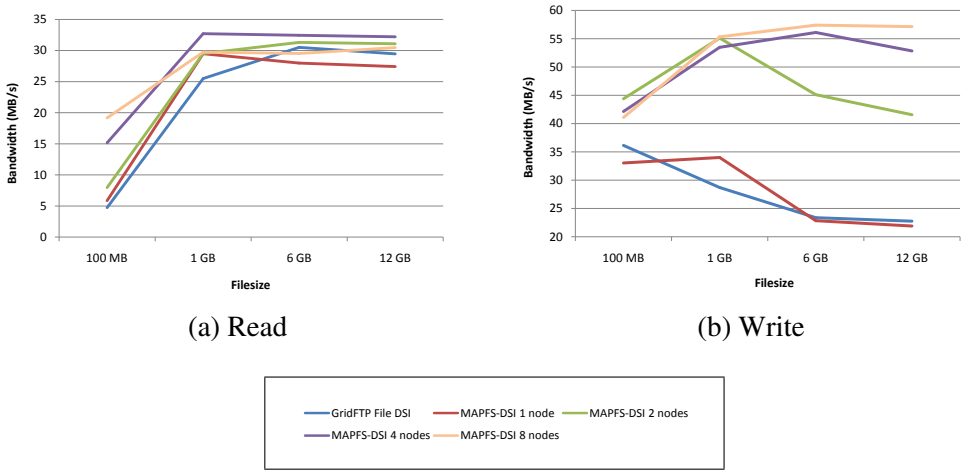
Fig. 8. Performance obtained by the proposed parallel WSRF-based data services to access a 12 GB filesize on *UPM 1* and *UPM 2*

the improvement is less than PDAS. However, different kinds of systems (such as MAPFS-DAI, fileAccess based on OGSA-DAI, CASTOR, and so on) could be used together by means of OGSA-DAI since they provide the same interface.

Furthermore, the performance of PDAS write operations achieves the maximum performance of the SOAP protocol used in data transfers of this kind of services. Nowadays, the maximum performance that can be achieved by SOAP is around 6-7 MB/s [43] due to different causes shown in [25]. The SOAP maximum performance limits the improvement of the *inter-cluster parallelism* in WSRF-based data services but the achievement of its maximum performance states the optimization of the use of the infrastructure thanks our proposal.

Secondly, we have compared the usual DSI provided by GridFTP, called *file* DSI, which only accesses the master cluster node, to our proposal, MAPFS-DSI, which accesses in a parallel way the whole cluster.

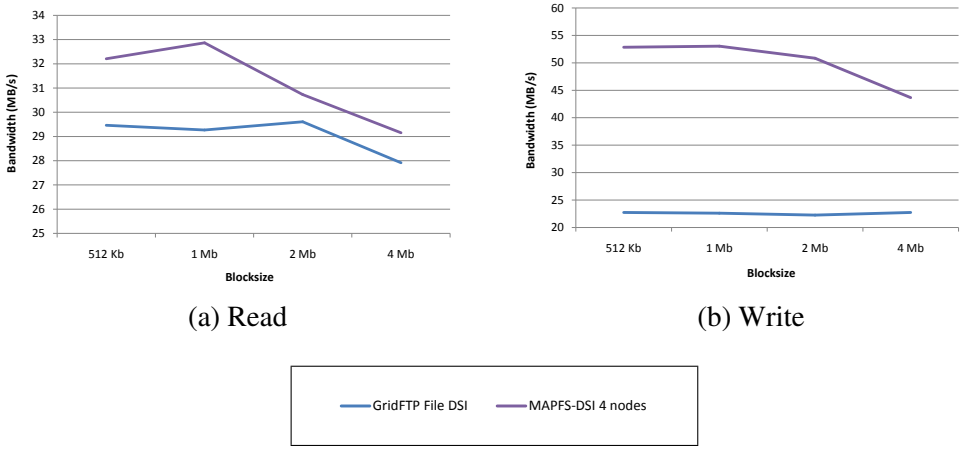
In this case, it is important to note that the time of accessing a file in a parallel way among the nodes of the cluster is superimposed by the transfer time. In this sense, as it can be seen in Figure 9, when the number of nodes increases, MAPFS-DSI provides an improvement vs. GridFTP file DSI, enhancing the use of GridFTP. Only if MAPFS-DSI is used without parallelism (1 node), its performance is lower than GridFTP file DSI because of the overhead observed in MAPFS when accessing a single node. Nevertheless, in some cases, the extra coordination overhead required by a higher number of nodes (8 nodes) cannot be compensated by the performance improvement.



(a) Read (b) Write

Fig. 9. Comparison between MAPFS-DSI and GridFTP file DSI to access a file on *UPM 1*

Furthermore, when the file size is increased, an increment of the performance is obtained, being higher in 1 GB. The reason behind this behavior is the influence of the memory hierarchy. Since *UPM 1* has 1 GB of RAM memory, most of data from 1 GB file size are accessed from memory instead of disk. On the other hand, file sizes higher than 1 GB require the access to disk, decreasing the performance. File sizes lower than 1 GB take advantage of the whole access to memory although the cost of the initial connection is high enough to hide the mentioned benefit. In this case, the low time to access the file size cannot make up for the high cost of the initial connection.



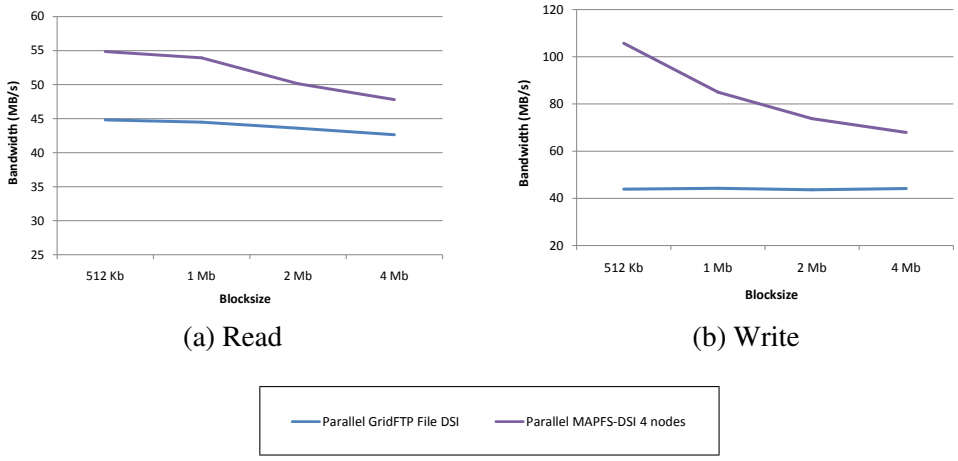
(a) Read (b) Write

Fig. 10. Comparison between MAPFS-DSI and GridFTP file DSI to access a file on *UPM 1* according to the blocksize

The block size has a meaningful influence in the performance of GridFTP and therefore MAPFS-DSI. According to block sizes, Figure 10 shows as block size is increased the obtained bandwidth decreases.

Moreover, both *inter-storage element* and *intra-cluster parallelism* can be considered. A parallel GridFTP *file* DSI version has been built to run tests, although it can only take advantage of *inter-storage element parallelism*. On the other hand, MAPFS-DSI can obtain benefits from both of them.

Figure 11 shows the I/O bandwidth obtained by the application of both levels of parallelism on *UPM 1* and *UPM 2* storage elements by using a version parallel of GridFTP *file* DSI and MAPFS-DSI.



(a) Read (b) Write

Fig. 11. Comparison between MAPFS-DSI and GridFTP file DSI to access a file on *UPM 1*

The use of both levels of parallelism obtains noteworthy improvements, as it can be seen in Figure 11. Increasing the number of storage elements and nodes of the cluster means an improvement in the performance. Whereas the average improvement achieved by the *inter-storage element parallelism* compared to a single storage element for read operations is 73.45 %, the improvement is 88.16 % for write operations. The *inter-storage parallelism* level also causes a great improvement in *file* DSI performance (*parallel GridFTP file-DSI*), but it is still worse than MAPFS-DSI.

Finally, Figure 11 shows as MAPFS-DSI achieves the maximum possible network bandwidth by using the two levels of parallelism. Considering that the work environment is a real environment and the network is not dedicated, such network is acting as a bottleneck of the whole system. Nevertheless, the performance obtained by *file* DSI of GridFTP is limited by the I/O bandwidth of the hard disks of the cluster master node. As the predicted advances in Computer Science state the network performance will increase higher than the I/O system [51], this is a much harder limitation.

All the shown results allow us to extract some interesting conclusions that validate our proposal. As grid environments are composed of different storage elements, and clusters stand out among them, it is possible to improve the grid data transfers accessing in a parallel way within clusters. Furthermore, a parallel access among several storage elements can enhance the whole transfer performance.

4 Related Work

In the last years, many projects have been performed to provide effective data access for grid applications. A taxonomy of data grids is proposed in [50]. The elements of this taxonomy related to store data are i) *data transport* and ii) *data replication and storage management*. Next subsections describe these two lines. At the end of this section, a comparison between current data grid approaches and our proposal is made.

4.1 Data transport

In data grid, data transport implies not only information transfer but also security and management issues. There are several commonly-used technologies on this field, which are going to be briefly commented.

SOAP [43] is a versatile protocol for exchanging messages over Internet. SOAP provides a basic and standard messaging framework especially suitable for Web services. Since the grid community has changed its orientation towards a service model [22] following the Web services approach, SOAP is considered as the default data transport in this model. However, SOAP is considerably slow and does not provide security, what must be solved by the middleware, for instance, by means of Grid Security Infrastructure (GSI) [24] in Globus.

GridFTP [1] is an extension of the standard FTP protocol, providing secure and efficient data management in grid architectures. In order to exhibit this behavior, GridFTP includes among others:

- Authentication, authorization, and access control by means of GSI.
- Parallel and striped data transfer.
- Support for reliable and restartable data transfer.
- Automatic negotiation of TCP buffer and window sizes.
- Instrumentation and monitoring tools.

The use of GridFTP parallel threads increases the effective network bandwidth improving the link utilization.

Alongside with these two basic, most-popular alternatives, there are many others aimed to provide more sophisticated data grid features. Internet Backplane Protocol (IBP) [42,8], Kangaroo [48] and Storage Resource Broker (SRB) [7] are examples of them.

The first one, IBP is an end-to-end data movement mechanism that tries to optimize the data movement from a node to another one by storing information at intermediate locations. IBP provides a semantic similar to UNIX system. It is based on a concept called exNode, similar to a UNIX inode.

Kangaroo is integrated into the Condor grid project and it presents itself as a reliable data movement system. It achieves reliability by running the transfer process in background and transparently managing faults. Kangaroo provides an interface to *get*, *put*, *commit*, and *push* operations.

The SRB I/O provides a UNIX-style file I/O interface for accessing heterogeneous data distributed over wide area nodes. SRB I/O provides streaming data transfer making multiple files possible to be sent using multiple streams to a storage resource.

4.2 Data replication and storage management

Due to the large-scale distributed nature of data grids not only information transfer, but also management (including data distribution, location, and replication) becomes a key element in providing efficient and reliable services. Nowadays there are several initiatives focused on these aspects, some of them integrated with the previously described data transfer technologies, some simply taking advantage of them and others based on their own solutions. The most significant examples are briefly described below.

GIGa-scale Global Location Engine (Giggle) [14] is a Replica Location Service (RLS) framework. The aim of RLS is to replicate data that are written once and read many. This service maintains information whereas there are physically located file replicas. A RLS is composed of a Replica Catalog and a Replica Location Index (RLI). Whereas the Replica Catalog is in charge of knowing the logical representa-

tion of the physical locations, the RLI indexes the catalog itself.

Grid Data Mirroring Package (GDMP) [44] is a secure and high-speed replication manager of data files and object databases by using Replica Catalogs and GridFTP. GDMP is based on the publisher-subscriber model. Every server publishes the set of files added to the Replica Catalog enabling clients request replicas of them. In order to provide security to the client connection, GSI is used.

SRB is one of the most widely used data grid architectures, since it provides a uniform interface to heterogeneous storage systems, unifying the view of the files stored in different locations. It uses replication to improve data availability and performance. The replication consistency in SRB is provided by means of synchronization and lock mechanisms propagating changes to other replicas.

Grid Datafarm (Gfarm) [46] is a high-speed I/O system designed for large-scale architectures. Although it can manage several petabytes, it requires high-speed network connections and large disk space in each storage element, which does not properly fit with the grid philosophy. In this way, it can be seen as adapted to the cluster computing idea.

The Storage Resource Managers (SRM) [38] interface provides a standard uniform management mechanism to heterogeneous storage systems. It provides a common interface to data grids, abstracting the peculiarities of each particular storage system. SRM could be used to access different storage system such as CERN Advanced STORage manager (CASTOR) [13], a scalable and distributed hierarchical storage management system developed at CERN in 1999. Other mass storage systems which provide a SRM interface are HPSS [53,29], Enstore [19], JASMine [26] and dCache [20,18].

OGSA-DAI [35,36] is a middleware designed to integrate data from different sources in grid environments. These include regular files, relational, and XML databases, etc. OGSA-DAI provides a set of standard Web services for clients to access information in a unified way.

4.3 Comparison between current data grid approaches and MAPFS-Grid

Current data grid approaches can be divided into three groups, according to the way in which the I/O operations are performed.

The first set of projects includes GridFTP [2], IBP [42], and Kangaroo [49], which can be considered as end-to end data-movement mechanisms that try to move data from a node to another. Once the data are in the local node, applications read or write to the new copy. This way of transfer is provided by MAPFS-DSI. The other two services of MAPFS-Grid, PDAS and MAPFS-DAI, enable performing direct read or writes to data.

The second set of projects, whose main examples are GASS [10], Legion I/O [54], and Gfarm [47], allows the application to perform direct operations on files, but what the system really does is a local copy of the file and then the application accesses it. This feature is one of the big differences from our work to these projects. Our system allows applications to directly read and write remote files, which are actually distributed within several storage elements.

The third group of projects where SRB [7], GridFS [11], CASTOR [12], OGSA-DAI [35] and XtreemFS [30] belong, offers the functionally of accessing remote files directly just like we do on MAPFS-Grid by means of PDAS and MAPFS-DAI. The difference with them, and also with all previous ones, is that our system takes advantage of two levels of parallelism that are not used by any of the projects presented so far. Most of them can only take advantage of one level of parallelism.

Finally, all previous projects offer only one interface, but MAPFS-Grid offers three skins for the user to be able to choose the most adequate one regarding compatibility vs. performance.

5 Conclusions and future work

The grid technology allows a large number of heterogeneous resources to be shared along geographically distributed sites. Data resources are common facilities in current large projects. Our proposal, MAPFS-Grid, provides a suite of three components for accessing efficiently large amounts of data in a grid environment. Every component is focused on a different aspect related to access data in grids. All of them take advantage of two levels of parallelism: the first level provides parallelism among all the storage resources of the grid and the second level optimizes the I/O operations in those storage resources corresponding to clusters.

Each MAPFS-Grid component fits properly with a different data scenario. MAPFS-DSI is useful for performance-critical applications since the obtained performance is optimal and it is only limited by the network bandwidth. MAPFS-Grid PDAS

and MAPFS-DAI are appropriate to service-oriented applications, which usually read and write parts of the remote files directly. Whereas MAPFS-Grid PDAS offers better performance, MAPFS-DAI provides interoperability. Therefore, in order to obtain a high number of accessible resources, MAPFS-DAI should be used. On the other hand, MAPFS-Grid PDAS is suitable for efficient service-oriented applications.

As a summary, this approach provides a high performance parallel grid storage system whose performance is only limited by the features of transfer methods, such as the transfer protocol or network bandwidth. These technological limitations restrict the benefits achieved by this work. Nevertheless, close advances in network technologies are going to involve a performance enhancement still high by this proposal. Some advantages of the use of the proposed high performance infrastructure are:

- Availability of huge shared storage capacity provided by several heterogeneous and scattered distributed resources. Besides, parallelism improves the use of the elements that make up the grid environment.
- Improvements of the I/O phase performance. Thus, MAPFS-Grid is especially beneficial for being used by HPC grid data-intensive applications.
- Use flexibility. The different functionalities of each MAPFS-Grid component make the adaptation to any kind of data access scenario possible.

Nevertheless, MAPFS-Grid also introduces a great level of complexity. The block-size parameter and the number of storage elements used in a parallel way have to be carefully chosen to reach the optimal performance. Furthermore, in a real environment where nodes are not dedicated to data storage, selecting the appropriate nodes is a challenging area. They must not only be chosen depending on their available disk space, but also considering their available bandwidth and current and future load. Thus, as ongoing and future work, we are working on the design of a complete system with all the benefits of the three components shown, which includes self-optimizing features with the aim of operating at optimal performance in any circumstances.

Acknowledgment

This work has been partially supported by the Spanish Ministry of Science and Technology under the TIN2007-60625 grant and one UPM pre-doctoral grant.

References

[1] B. Allcock, J. Bester, J. Bresnahan, A. L. Chervenak, I. Foster, C. Kesselman, S. Meder, V. Nefedova, D. Quesnal, and S. Tuecke. Data management and transfer in high performance computational grid environments. *Parallel Computing Journal*, 28(5):749–771, May 2002.

[2] W. Allcock, J. Bester, J. Bresnahan, A. Chervenak, L. Liming, S. Meder, and S. Tuecke. GridFTP Protocol Specification, September 2002.

[3] W. Allcock, J. Bresnahan, R. Kettimuthu, and M. Link. The Globus Striped GridFTP Framework and Server. In *Proceedings of the 2005 ACM/IEEE conference on Supercomputing (SC '05)*, page 54, Washington, DC, USA, 2005. IEEE Computer Society.

[4] M. Antonioletti, M. Atkinson, R. Baxter, A. Borley, N. Chue Hong, P. Dantressangle, A. Hume, M. Jackson, A. Krause, S. Laws, P. Parson, N. Paton, J. Schopf, T. Sugden, P. Watson, and D. Vyvyan. OGSA-DAI status and benchmarks. In *Proceeedings of UK e-Science All Hands Meeting*, 2005.

[5] M. Atkinson, K. Karasavvas, M. Antonioletti, R. Baxter, A. Borley, N. Chue Hong, A. Hume, M. Jackson, A. Krause, S. Laws, N. Paton, J. Schopf, T. Sugden, K. Tourlas, and P. Watson. A new architecture for OGSA-DAI. *AHM*, 2005.

[6] P. Avery. Data grids: a new computational infrastructure for data-intensive science. *Philosophical Transactions of the Royal Society of London Series a-Mathematical Physical and Engineering Sciences*, 360(1795):1191–1209, 2002.

[7] C. K. Baru, R. W. Moore, A. Rajasekar, and M. Wan. The SDSC storage resource broker. In S. A. MacKay and J. H. Johnson, editors, *Proceedings of the 1998 conference of the Centre for Advanced Studies on Collaborative Research (CASCON)*, page 5, Toronto, Ontario, Canada, December 1998. IBM Press.

[8] A. Bassi, M. Beck, T. Moore, J. S. Plank, M. Swany, R. Wolski, and G. Fagg. The Internet Backplane Protocol: a study in resource sharing. *Future Generation Computer Systems*, 19(4):551–562, 2003.

[9] Daniel Becker, Wolfgang Frings, and Felix Wolf. Performance evaluation and optimization of parallel grid computing applications. In *Proceedings of the 16th Euromicro Conference on Parallel, Distributed and Network-Based Processing (PDP 2008)*, pages 193–199, Washington, DC, USA, 2008. IEEE Computer Society.

[10] Joseph Bester, Ian Foster, Carl Kesselman, Jean Tedesco, and Steven Tuecke. Gass: a data movement and access service for wide area computing systems. In *IOPADS '99: Proceedings of the sixth workshop on I/O in parallel and distributed systems*, pages 78–88, New York, NY, USA, 1999. ACM Press.

[11] Dheeraj Bhardwaj and Manish Sinha. Gridfs: Ensuring high-speed data transfer using massively parallel i/o. In Subhash Bhalla, editor, *DNIS*, volume 3433 of *Lecture Notes in Computer Science*, pages 280–287. Springer, 2005.

[12] CASTOR - CERN Advanced STORage manager, accessed 2008 [online]. available: *http://castor.web.cern.ch/castor*.

[13] CASTOR - CERN Advanced STORage manager, accessed Apr 2007 [Online]. Available: *http://castor.web.cern.ch/castor*.

[14] A. Chervenak, E. Deelman, I. Foster, L. Guy, W. Hoschek, A. Iamnitchi, C. Kesselman, P. Kunszt, M. Ripeanu, B. Schwartzkopf, H. Stockinger, K. Stockinger, and B. Tierney. Giggle: a framework for constructing scalable replica location services. In *Proceedings of the 2002 ACM/IEEE conference on Supercomputing (Supercomputing '02)*, pages 1–17, Los Alamitos, CA, USA, 2002. IEEE Computer Society Press.

[15] A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, and S. Tuecke. The Data Grid: Towards an architecture for the distributed management and analysis of large scientific datasets. *Journal of Network and Computer Applications*, 23(3):187–200, July 2000.

[16] K. Chiu, M. Govindaraju, and R. Bramley. Investigating the Limits of SOAP Performance for Scientific Computing. In *Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing (HPDC '02)*, page 246, Washington, DC, USA, 2002. IEEE Computer Society.

[17] Data Access and Integration Services working group, accessed May 2007 [Online]. Available: *https://forge.gridforum.org/projects/dais-wg*.

[18] dCache.ORG, accessed Apr 2007 [Online]. Available: *http://www.dcache.org/*.

[19] Fermilab MASS STORAGE SYSTEM - Enstore, accessed Apr 2007 [Online]. Available: *http://www-ccf.fnal.gov/enstore/*.

[20] M. Ernst, P. Fuhrmann, M. Gasthuber, T. Mkrtchyan, and C. Waldman. dCache, a Distributed Storage Data Caching System. In *Proceedings of the International Conference on Computing on High-Energy and Nuclear Physics (CHEP'01)*, page 244, Beijing, China, September 2001.

[21] I. Foster et al. The Open Grid Services Architecture, version 1.0. 2005.

[22] I. Foster, C. Kesselman, J. M. Nick, and S. Tuecke. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. Published online at *http://www.globus.org/research/papers/ogsa.pdf*, January 2002.

[23] Lori A. Freitag and Raymond M. Loy. Adaptive, multiresolution visualization of large data sets using a distributed memory octree. In *Proceedings of SC99: High Performance Networking and Computing*, Portland, OR, November 1999. ACM Press and IEEE Computer Society Press.

[24] Grid Security Infrastructure (GSI), accessed April 2007 [Online]. Available: *http://www.globus.org/toolkit/docs/4.0/security*.

[25] E. Gómez-Martínez and J. Merseguer. Impact of SOAP Implementations in the Performance of a Web Service-Based Application. In Geyong Min, Beniamino Di Martino, Laurence Tianruo Yang, Minyi Guo, and Gudula Rünger, editors, *Frontiers of High Performance Computing and Networking, ISPA 2006 International Workshops*, volume 4331 of *Lecture Notes in Computer Science*, pages 884–896. Springer, 2006.

[26] B. K. Hess, M. Haddox-Schatz, and M. A. Kowalski. The Design and Evolution of Jefferson Lab's Jasmine Mass Storage System. In *Proceedings of the 22nd IEEE / 13th NASA Goddard Conference on Mass Storage Systems and Technologies (MSST'05)*, pages 94–105. IEEE Computer Society, 2005.

[27] K. Holtman. Object Level Physics Data Replication in the Grid. In *Proceedings of the VII International Workshop on Advanced Computing and Analysis Techniques in Physics Research (ACAT 2000)*, pages 244–246, October 2000.

[28] W. Hoschek, F. J. Jaen-Martinez, A. Samar, H. Stockinger, and K. Stockinger. Data management in an international data grid project. In *Proceedings of the First IEEE/ACM International Workshop on Grid Computing(GRID 2000)*, December 2000.

[29] HPSS - High Performance Storage System, accessed Apr 2007 [Online]. Available: *http://www.hpss-collaboration.org/hpss/*.

[30] F. Hupfeld, T. Cortes, B. Kolbeck, E. Focht, M. Hess, J. Malo, J. Marti, J. Stender, and E. Cesario. Xtreemfs: a case for object-based storage in grid data management. In *Proceedings of 33th International Conference on Very Large Data Bases (VLDB) Workshops*, 2007.

[31] Konstantinos Karasavvas, Mario Antonioletti, Malcolm P. Atkinson, Neil P. Chue Hong, Tom Sugden, Alastair C. Hume, Mike Jackson 0003, Amrey Krause, and Charaka Palansuriya. Introduction to ogsa-dai services. In Pilar Herrero, María S. Pérez, and Víctor Robles, editors, *SAG*, volume 3458 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2004.

[32] Thilo Kielmann, Henri E. Bal, Jason Maassen, Rob van Nieuwpoort, Lionel Eyraud, Rutger Hofman, and Kees Verstoep. Programming environments for high-performance grid computing: the Albatross project. *Future Generation Computer Systems*, 18(8):1113–1125, 2002.

[33] H. Liefke and D. Suciu. XMILL: An Efficient Compressor for XML Data. In W. Chen, J. F. Naughton, and P. A. Bernstein, editors, *SIGMOD Conference*, pages 153–164. ACM, 2000.

[34] P. M. Lyster, C. H. Q. Ding, K. Ekers, R. Ferraro, J. Guo, M. Harber, D. Lamich, J. W. Larson, R. Lucchesi, R. Rood, S. Schubert, W. Sawyer, M. Sienkiewicz, A. da Silva, J. Stobie, L. L. Takacs, R. Todling, and J. Zero. Parallel computing at the nasa data assimilation office (dao). In *Proceedings of the 1997 ACM/IEEE conference on Supercomputing (Supercomputing '97:)*, pages 1–18, New York, NY, USA, 1997. ACM.

[35] J. Magowan. A view on relational data on the grid. In *Proceedings of the 17th International Symposium on Parallel and Distributed Processing (IPDPS '03)*, page 90, Washington, DC, USA, 2003. IEEE Computer Society.

[36] Susan Malaika, Andrew Eisenberg, and Jim Melton. Standards for databases on the grid. *SIGMOD Record*, 32(3):92–100, 2003.

[37] OASIS. Ws-ResourceFramework, accessed 2009. [Online]. Available: http://www.oasis-open.org. 2005.

[38] T. Perelmutov, D. Petravick, E. Corso, L. Magnoni, O. Barring, J. Baud, F. Donno, M. Litmaath, S. De Witt, J. Jensen, M. Haddox-Schatz, B. Hess, A. Kowalski, and C. Watson. The storage resource manager interface specification, version 2.2. Lawrence Berkeley National Laboratory, January 2007.

[39] María S. Pérez, Jesús Carretero, José M. Peña Félix García, and Víctor Robles. MAPFS: A flexible multiagent parallel file system for clusters. *Future Generation Computer Systems*, 22(5):620–632, 2006.

[40] María S. Pérez, Jesús Carretero, Félix García, José M. Peña Sánchez, and Victor Robles. MAPFS-Grid: A flexible architecture for data-intensive grid applications. In F. Fernández Rivera, Marian Bubak, A. Gómez Tato, and Ramon Doallo, editors, *European Across Grids Conference*, volume 2970 of *Lecture Notes in Computer Science*, pages 111–118. Springer, 2003.

[41] María S. Pérez, Félix García, and Jesús Carretero. A new multiagent based architecture for high performance I/O in clusters. In *ICPP Workshops*, pages 201–206. IEEE Computer Society, 2001.

[42] J. S. Plank, M. Beck, W. Elwasif, T. Moore, M. Swany, and R. Wolski. The Internet Backplane Protocol: Storage in the network. In *NetStore '99: Network Storage Symposium*. Internet2, *http://dsi.internet2.edu/netstore99*, October 1999.

[43] SOAP Version 1.2: Messaging Framework, accessed Jun 2007 [Online]. Available: *http://www.w3.org/TR/soap12-part1/*, April 2007.

[44] H. Stockinger, A. Samar, S. Muzaffar, and F. Donno. Grid Data Mirroring Package (GDMP). *Scientific Programming*, 10(2):121–133, 2002.

[45] N. Sundaresan and R. Moussa. Algorithms and programming models for efficient representation of XML for Internet applications. *Computer Networks*, 39(5):681–697, 2002.

[46] O. Tatebe, Y. Morita, S. Matsuoka, N. Soda, and S. Sekiguchi. Grid datafarm architecture for petascale data intensive computing. In *Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID '02)*, page 102, Washington, DC, USA, 2002. IEEE Computer Society.

[47] Osamu Tatebe, Youhei Morita, Satoshi Matsuoka, Noriyuki Soda, and Satoshi Sekiguchi. Grid datafarm architecture for petascale data intensive computing. In *CCGRID '02: Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid*, page 102, Washington, DC, USA, 2002. IEEE Computer Society.

[48] 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.

[49] Douglas Thain, Jim Basney, Se-Chang Son, and Miron Livny. The kangaroo approach to data movement on the grid. *HPDC*, 00:0325, 2001.

[50] S. Venugopal, R. Buyya, and K. Ramamohanarao. A taxonomy of Data Grids for distributed data sharing, management, and processing. *ACM Computing Surveys*, 38(1):3, 2006.

[51] C. Vilett. Moore's law vs. storage improvements vs. optical improvements. *Scientific American*, January 2001.

[52] W3C. SOAP Message Transmission Optimization Mechanism, accessed 2009 [Online]. Available: *http://www.w3.org/TR/soap12-mtom*, January 2005.

[53] R. W. Watson and R. A. Cyne. The Parallel I/O Architecture of the High-Performance Storage System (HPSS). In *Proceedings of the IEEE Symposium on Mass Storage Systems*, pages 27–44, 1995.

[54] Brian S. White, Andrew S. Grimshaw, and Anh Nguyen-Tuong. Grid-based file access: The Legion I/O model. In *HPDC*, pages 165–174, 2000.

[55] S.J. Young, G.Y. Fan, D. Hessler, S. Lamont, T.T. Elvins, M. Hadida, G. Hanyzewski, J.W. Durkin, P. Hubbard, G. Kindlmann, E. Wong, D. Greenberg, S. Karin, and M.H. Ellisman. Implementing a collaboratory for microscopic digital anatomy. *Supercomputer Applications and High Performance Computing*, 10(2/3):170–181, 1996.