The Open Grid Services Architecture and Data Grids

Peter Z. Kunszt and Leanne P. Guy
IT Division - Database Group, CERN, 1211 Geneva Switzerland

July 7th, 2002

Abstract

Data Grids address the data intensive aspects of Grid computing, and therefore impose a very specific set of requirements on Grid Services. In this article the Data Grid problem is revisited with emphasis on Data Management. It investigates how Data Grid Services would need to be deployed within the Open Grid Services Architecture to fulfill the vision of the Data Grid.

1 Introduction

Data Grids address computationally and data intensive applications that combine very large datasets and a wide geographical distribution of users and resources [1], [13]. In addition to computing resource scheduling, Data Grids address the problems of storage and data management, network-intensive data transfers and data access optimization, while maintaining high reliability and availability of the data (see [13], [14] and references therein).

The Open Grid Services Architecture (OGSA) [1], [3] builds upon the Anatomy of the Grid [6], where the authors present an open Grid architecture and define the technologies and infrastructure of the Grid as “supporting the sharing and coordinated use of diverse resources in dynamic distributed Virtual Organizations (VOs)”. OGSA extends and complements the definitions given in [6] by defining the architecture in terms of Grid Services and by aligning it with emerging Web Service technologies.

Web Services are an emerging paradigm in distributed computing that focus on simple standards-based computing models. OGSA picks the Web Service Description Language (WSDL) [7], the Simple Object Access Protocol (SOAP) [8] and the Web Service Introspection Language (WSIL) [9] from the set of technologies offered by Web Services and capitalizes especially on WSDL. This is a very natural approach since in the distributed world of Data Grid Computing the same problems arise as on the Internet concerning the description and discovery of services, and especially the heterogeneity of data [5].

In this article the application of the Open Grid Services Architecture is discussed with respect to Data Grids. We revisit the problem that is being addressed, the vision and functionality of Data Grids and of OGSA. We then investigate OGSA's benefits and possible weaknesses in view of the Data Grid problem. In

conclusion, we address what we feel are some of the shortcomings and open issues that expose potential areas of future development and research.

The European Data Grid project (EU DataGrid project) aims to build a computational and data intensive grid to facilitate the analysis of millions of Gigabytes of data generated by the next generation of scientific exploration. The main goal is to develop an infrastructure to will allow all scientists participating in a research project, the best possible access to data and resources, irrespective of geographical location. All of the Data Grid discussion presented in this paper draws from existing concepts in the literature, but is strongly influenced by the design and experiences with data management in the context of EU DataGrid project [30]. In particular, data management in the EU DataGrid project has focused primarily on file based data management [33]. However, the usage of the Web Services paradigm to build a Grid infrastructure was always considered in the architecture and has lead to the definition of the Web Service Discovery Architecture WSDA [4].

1.1 The Vision

In this section we summarize the Data Grid problem and the vision of the Grid as presented in the literature [1], [6], [2].

1.2 Desirable features of Grids

In the scientific domain there is a need to share resources to solve common problems and to facilitate collaboration among many individuals and institutes. Likewise, in commercial enterprise computing there is an increasing need for resource sharing, not just to solve common problems, but to enable enterprise-wide integration and business to business (B2B) partner collaboration

To be successful on a large scale, the sharing of resources should be:

FLEXIBLE, SECURE, COORDINATED, ROBUST, SCALABLE,

UBIQUITOUSLY ACCESSIBLE, MEASURABLE (QOS METRICS),

TRANSPARENT TO THE USERS

The distributed resources themselves should have the following qualities:

INTEROPERABLE, MANAGEABLE, AVAILABLE AND EXTENSIBLE

Grid computing attempts to address this very long list of desirable properties. The concept of the Virtual Organization (VO) gives us the first necessary semantics by which to address these issues systematically and to motivate these properties in detail.

1.3 Virtual Organizations

A Virtual Organization (VO) is defined as a group of individuals or institutes who are geographically distributed but who appear to function as one single unified organization. The members of a VO usually have a common focus, goal or vision, be it a scientific quest or a business venture. They collectively dedicate

resources, for which a well-defined set of rules for sharing and QoS exist, to this end. Work is coordinated through electronic communication. The larger the VO, the more decentralised and geopraphically distributed it tends to be in nature.

The concept of a Virtual Organization is not a new one; VOs have existed for many years in the fields of business and engineering, where consortia are assembled to bid for contracts, or small independent companies temporarily form a VO to share skills, reduce costs and expand their market access. In the commercial context, VOs may be set up for anything from a single enterprise entity to complex B2B relationships.

In scientific communities such as High Energy Physics (HEP), Astrophysics or Medical research, scientists form collaborations, either individually or between their respective institutes, to work on a common set of problems, or to share a common pool of data. The term Virtual Organization has only recently been coined to describe such collaborations in the scientific domain.

VOs may be defined in a very flexible way so as to address highly specialized needs and are instantiated through the set of services that they provide to their users/applications. Ultimately, the organizations, groups and individuals that form a VO also need to pay for the resources that they access in some form or another – either by explicitly paying money for services provided by Grid service providers or by trading their own resources. Grid resource and service providers may choose to which VOs they offer their resources and services. The models for paying and trading of resources within a VO will probably be vastly different in the scientific and business domains.

In the vision of the Grid, end users in a VO can use the shared resources according to the rules defined by the VO and the resource providers. Different VOs may or may not share the same resources. VOs can be dynamic, i.e. they may change in time both in scope and extension by adjusting the services that are accessible by the VO and the constituents of the VO. The Grid needs to support these changes such that they are transparent to the users in the VO.

1.4 Motivation for the Desirable Features in Data Grids

In the specific case of Data Grids, the members of a VO also share their collective data, which is potentially distributed over a wide area (SCALABLE). The data needs to be accessible from anywhere at anytime (FLEXIBLE, UBIQUITOUS ACCESS) however the user is not necessarily interested in the exact location of the data (TRANSPARENT). The access to the data has to be secured, enabling different levels of security according to the needs of the VO in question, while at the same time maintaining the manageability of the data and its accessibility (SECURE, COORDINATED, MANAGEABLE). The reliability and robustness requirements in the existing communities interested in Data Grids are high, since the computations are driven by the data – if the data are not accessible, no computation is possible (ROBUST, AVAILABLE).

In many VOs, there is a need to access existing data – usually through interfaces to legacy systems and data stores. The users do not want to be concerned with issues of data interoperability and data conversion (INTEROPERABLE, EXTENSIBLE). In addition, some Data Grids also introduce the concept of Virtual Data [20] where data needs to be generated by a series of computations before it is accessible to the user.

2 The OGSA Approach

The Open Grid Services Architecture views the Grid as an extensible set of Grid Services. A Grid Service is defined as "a Web Service that provides a set of well-defined interfaces and that follows specific conventions" [2]. Virtual Organizations are defined by their associated services, since the set of services 'instantiate' a VO. In OGSA each VO builds its infrastructure from existing Grid Service components and has the freedom to add custom components that have to implement the necessary interfaces to qualify as a Grid Service. We summarize the requirements that motivate the OGSA as stated in [1] and [3] and the properties it is providing to the Grid (see previous section for the discussion of properties) in the table below:

	Requirement	***Property***
1	Support the creation, maintenance and application of ensembles of services maintained by VOs.	COORDINATED, TRANSPARENT, SCALABLE, EXTENSIBLE, ***MANAGEABLE***
2	Support local and remote transparency with respect to invocation and location.	***TRANSPARENT***, FLEXIBLE

3	Support multiple protocol bindings (enabling e.g. localized optimization and protocol negotiation).	***FLEXIBLE***, TRANSPARENT, INTEROPERABLE, EXTENSIBLE, UBIQ.ACCESSIBLE
4	Support Virtualization in order to provide a common interface encapsulating the actual implementation.	***INTEROPERABLE***, TRANSPARENT, MANAGEABLE
5	Require mechanisms enabling interoperability.	***INTEROPERABLE,*** UBIQ.ACCESSIBLE
6	Require standard semantics (like same conventions for error notification).	***INTEROPERABLE***
7	Support transience of services.	SCALABLE, MANAGEABLE, AVAILABLE, ***FLEXIBLE***
8	Support upgradeability without disruption of the services.	MANAGEABLE, FLEXIBLE, ***AVAILABLE***

Table 1: Requirements mentioned in [2],[3] for OGSA. We try to map each requirement to the properties that Grid Systems are supposed to have, based on the requirements given in Section 2. The primary property is highlighted. An analysis of these requirements is given in the next section.

As mentioned before, OGSA focuses on the nature of services that makes up the Grid. In OGSA existing Grid technologies are aligned with Web Service technologies in order to profit from the existing capabilities of Web Services, including

- Service description and discovery
- Automatic generation of client and server code from service descriptions
- Binding of service descriptions to interoperable network protocols
- Compatibility with higher-level open standards, services and tools
- Broad industry support

OGSA thus relies upon emerging Web Service technologies to address the requirements listed in Table 1. OGSA defines a preliminary set of Grid Service interfaces and capabilities that can be built upon. All of these interfaces are defined using WSDL to ensure a distributed service system based on standards. They are designed such that they address the following responsibilities:

Responsibility	***Interfaces***	***Requirements addressed***
Information & Discovery	GridService.FindServiceData Registry.(Un)RegisterService HandleMap.FindByHandle *Service Data Elements* *Grid Service Handle and Grid Service Reference*	1,2,4,5,6
Dynamic service creation	Factory.CreateService	1,7
Lifetime Management	GridService.SetTerminationTime, GridService.Destroy	1,7
Notification	NotificationSource. (Un)SubscribeToNotificationTopic NotificationSink.DeliverNotification	5
Change Management	*CompatibilityAssertion data element*	8
Authentication	Impose on the Hosting Environment	SECURE
Reliable Invocation	Impose on Hosting Environment	ROBUST
Authorization and policy management	Defer to later	SECURE
Manageability	Defer to later	MANAGEABLE

Table 2: Current elements of OGSA and elements that have been mentioned by OGSA but are said to be dealt with elsewhere. We also refer to the requirement or to the Grid capability that the elements address. Please be aware that this is a snapshot and that the current OGSA service specification might not be reflected correctly.

The GridService interface is imposed on all services. The other interfaces are optional, although there need to be many service instances present in order to be able to run a VO. Based on the services defined, higher-level services are envisaged such as data management services, workflow management, auditing, instrumentation and monitoring, problem determination and security protocol mapping services. This is a non-exhaustive list and is expected to grow in the future.

OGSA introduces the concept of Grid Service Handles that are unique to a service and by which the service may be uniquely identified and looked up in the HandleMap. Grid Service Handles are immutable entities. The Grid Service Reference is then the description of the service instance in terms of (although not necessarily restricted to) WSDL and gives the user the possibility to refer to a

mutable running instance of this service. OGSA envisages the possibility of a hierarchical structure of service hosting environments, as they demonstrate in their examples.

Another essential building block of OGSA is the standardized representation for service data, structured as a set of named and typed XML fragments called *service data elements* (SDE). The SDE may contain any kind of information on the service instance allowing for basic introspection information on the service. The SDE is retrievable through the GridService port type's FindServiceData operation implemented by all Grid Services. Through the SDE the Grid Services are stateful services, unlike Web Services that do not have a state. This is an essential difference between Grid Services and Web Services: Grid Services are stateful and provide a standardized mechanism to retrieve their state.

3 Data Grid Services

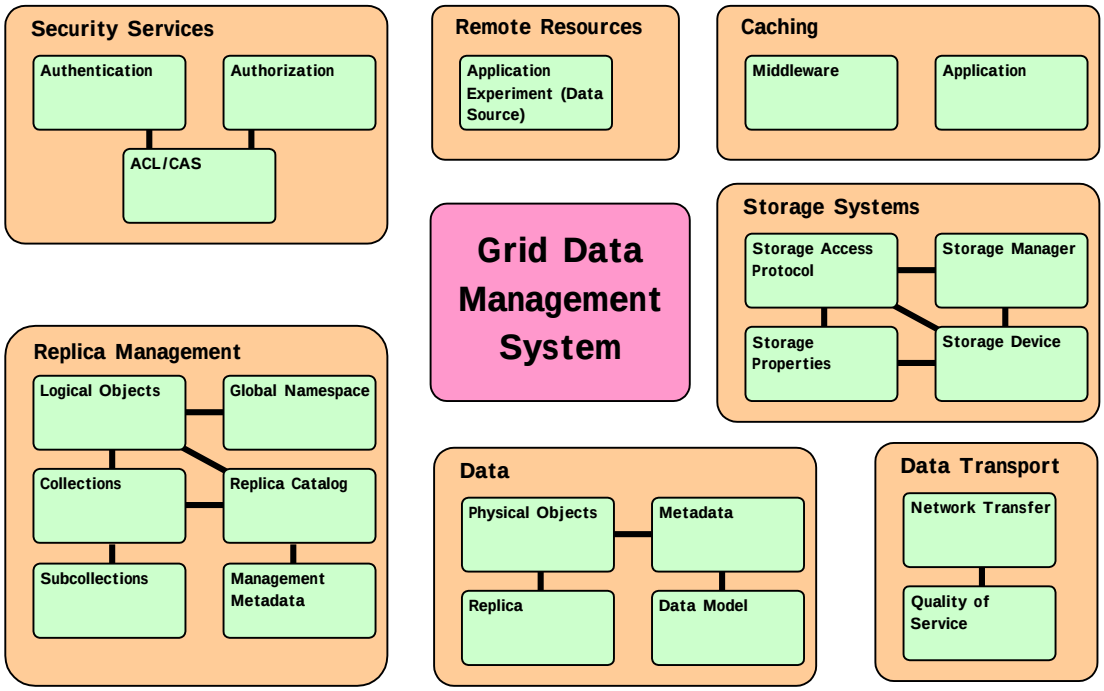

Figure 1: The concept space of the GGF Data Area (from the GGF Data Area website)

There has been a lot of effort put into the definition of Data Grid services in the existing Grid projects. Figure 1 shows the Global Grid Forum GGF Data Area concept space, i.e. their domains of interest. In this article we address the same concept space. Examples of Data Grid architectures are given in [13,17-23].

3.1.1 The Data

In most of the existing architectures the data management services are restricted to the handling of files. However, for many VOs files represent only an intermediate level of data granularity. In principle, Data Grids need to be able to handle data elements – from single bits to collections of collections and even virtual data, which must be generated upon request. All kinds of data need to be identifiable through some mechanism – a logical name or ID – that in turn can be used to locate and access the data. This concept is identical to the concept of a Grid Service Handle in OGSA [2] so in order to be consistent with the OGSA terminology we will call this logical identifier the Grid Data Handle GDH (see below).

The following is a non-definitive list of kinds of data that are dealt with in Data Grids.

- **Files**. For many of the VOs the only access method to data is file based I/O, data is kept only in files and there is no need for other kind of data granularity. This simplifies data management in some respect because the semantics of files are well understood. For example, depending on the QoS requirements on file access the files can be secured through one of the many known mechanisms (Unix permissions, ACLs, etc). Files can be maintained through directory services like the Globus Replica Catalog [25], which maps the logical file name to their physical instances, and other metadata services that are discussed below.
- **File Collections**. Some VOs want to have the possibility of assigning logical names to file collections that are recognized by the Grid. Semantically there are two different kinds of file collections: *confined* collections of files where all files making up the collection are always kept together and are effectively treated as one – just like a *tar* or *zip* archive – and *free* collections that are composed of files as well as other collections not necessarily available on the same resource – they are effectively a bag of logical file and collection identifiers. Confined collections assure the user that all the data are always accessible at a single data source while free collections provide a higher flexibility to freely add and remove items from the collection but don't guarantee that all members are accessible or even valid at all times. Keeping free collections consistent with the actual files requires additional services.
- **Relational Databases**. The semantics of data stored in a Relational Database Management System (RDBMS) are also extremely well understood. The data identified by a GDH may correspond to a database, a table, a view or even to a single row in a table of the RDBMS. Distributed database access in the Grid is an active research topic and is being treated by a separate article in this volume [24].
- **XML Databases and Semistructured Data**. Data with loosely defined or irregular structure is best represented using the semistructured data model. It is essentially a dynamically typed data model that allows a

"schema-less" description format in which the data is less constrained than in relational databases [32]. The XML format is the standard in which the semistructured data model is represented. A GDH may correspond to any XML data object identifiable in an XML Database.

- **Data Objects**. This is the most generic form of a single data instance. The structure of an object is completely arbitrary, so the Grid needs to provide services to describe and access specific objects since the semantics may vary from object type to object type. VOs may choose different technologies to access their objects – ranging from proprietary techniques to open standards like CORBA/IIOP and SOAP/XML.
- **Virtual Data**. The concept of virtualized data is very attractive to VOs that have large sets of secondary data that are derived of primary data using a well defined set of procedures and parameters. If the secondary data are more expensive to store than to regenerate, they can be virtualized i.e. only created upon request. Additional services are required to manage virtual data.
- **Data Sets**. Data sets differ from free file collections only in that they can contain any kind of data from the list above in addition to files. Such data sets are useful for archiving, logging and debugging purposes. Again, the necessary services that track the content of data sets and keep them consistent need to be provided.

3.1.2 The Grid Data Handle GDH

The common requirement for all these kinds of data is that the logical identifier of the data, the Grid Data Handle (GDH), be globally unique. The GDH can be used to locate the data and retrieve information about it. It can also be used as a key by which more attributes can be added to the data through dedicated application-specific catalogs and metadata services. There are many possibilities to define a GDH, the simplest being the well-known UUID scheme [28]. In order to be humanly readable, a URI-like string might be composed, specifying the kind of data instead of the protocol, a VO namespace instead of the host and then a VO-internal unique identifier, which may be a combination of creation time and creation site and a number (in the spirit of UUID), or another scheme that may even be set up by the VO. The assignment and the checking of the GDH can be performed by the Data Registry (see below).

Just like for the GSH, a semantic requirement on the GDH is that it must not change over time once it has been assigned. The reason for this requirement is that we want to enable automated Grid services to be able to perform all possible operations on the data – like location, tracking, transmission, etc. This can easily be achieved if we have a handle by which the data is identified. One of the fundamental differences between Grids and the Web is that for Web services ultimately there always is a human being at the end of the service chain who can take corrective action if some of the services are erroneous. On the Grid this is not the case anymore. If the data identifier is neither global nor unique, tracking

by automated services will be very difficult to implement – they would need to be as 'smart' as humans in identifying the data based on its content.

3.1.3 The Grid Data Reference

For reasons of performance enhancement and scalability, a single logical data unit (any kind) identified by a GDH may have several physical instances – replicas – located throughout the Grid, all of which also need to be identified. Just like the GSH, the GDH carries no protocol- or instance-specific information such as supported protocols to access the data. All of this information may be encapsulated into a single abstraction – in analogy to the Grid Service Reference (GSR) – into a Grid Data Reference GDR. Each physical data instance has a corresponding GDR that describes how the data can be accessed. This description may be in WSDL, since a data item also can be viewed as a Grid resource. The elements of the GDR include physical location, available access protocols, data lifetime, and possibly other metadata such as size, creation time, last update time, created by, etc. The exact list and syntax would need to be agreed upon but this is an issue for the standardization efforts in bodies like GGF. The list should be customizable by VOs because items like update time may be too volatile to be included in the GDR metadata.

In the OGSA context, the metadata may be put as Service Data Elements in the Data Registry.

3.1.4 The Data Registry

The mapping from GDH to GDR is kept in the Data Registry. This is the most basic service for Data Grids, which is essential to localize, discover and describe any kind of data. The Service Data Elements of this service will hold most of the additional information on the GDR to GDH mapping, but there may be other more dedicated services for very specific kinds of GDH or GDR metadata.

To give an example, in Data Grids that only deal with files – as is currently the case within the EU DataGrid project – the GDH corresponds to the Logical File Name LFN and the GDR is a combination of the physical file name and the transport protocol needed to access the file, sometimes also called Transport File Name TFN [17-19]. The Data Registry corresponds to the Replica Catalog [26] in this case.

3.2 The Functionality and the Services

In this section we discuss the functionalities of Data Grids that are necessary and/or desirable in order to achieve the Data Grid vision.

3.2.1 VO Management

Virtual Organizations can be set up, maintained and dismantled by Grid Services. In particular there needs to be functionality provided for bootstrapping, user management, resource management, policy management and budget management. However, these functionalities are generic to all kinds of Grids and their detailed discussion is beyond the scope of this article. Nevertheless it is

important to note that since VOs form the organic organizational unit in Grids, VO management functionalities need to be supported by all services. Some of these issues are addressed by the services for control and monitoring, see below.

3.2.2 Data Transfer

The most basic service in Data Grids is the data transfer service. To ensure reliable data transfer and replication, data validation services need to be set up. Data validation after transfer is a QoS service that might not be required by all VOs. For file-based Data Grids there is already an on-going effort to provide a reliable file transfer service for the Grid [25].

3.2.3 Data Storage

Data Grid storage services will be provided by many resources. There will be very primitive storage systems with limited functionality as well as high level systems for data management such as RDBMSs or Storage Resource Managers [19]. All may provide a high quality Grid Services, but with vast range of available functionality. The requirements on the data store depend very much on the kind of data (see section 3.1.1). Two common QoS additions are:

Resource allocation: The ability to reserve directly (through advance reservation) or indirectly (through quotas) sufficient space for data to be transferred to the store later.

Lifetime management: The ability to specify the lifetime of the data directly (timestamps and expiration times), or indirectly (through data status flags like permanent, durable, volatile).

3.2.4 Data Management

In all existing Data Grids data management is one of the cornerstones of the architecture. The data management services need to be very flexible in order to accommodate the peculiarities and diverse requirements on quality of service of the VOs and their peculiarities with respect to different kinds of data (see previous section) and data access.

In the vision of a Data Grid, the data management services maintain, discover, store, validate, transfer, instantiate the data for the user's applications transparently. Because the multitude of different kinds of data it is essential that the data can be described and validated in the Data Grid framework.

In the previous section we introduced the concept of Grid Data Handles and Grid Data References as well as a service for their mapping, the Data Registry. These are the core concepts of data management. In addition, we have the following data management functionalities in Data Grids:

Registration of data: The first obvious functionality is to register new or existing data on the Grid. The data will receive a GDH; first instances will have to be described through their GDR, and the corresponding mappings need to be registered in the Data Registry (DR). This functionality may be part of the DR or set up as a separate data registration service.

Materialization of virtual data: This functionality of the data management services will materialize virtual data according a set of materialization instructions. These instructions include references to the executables and any input datasets required, as well as to any additional job parameters. All informantion pertaining to the materialization of virtual data is stored in a catalog. After materialization, physical copies of the data exist and the corresponding catalogs need to be updated and a GDR assigned.

GDH assignment and validation: The uniqueness of the GDH can only be assured by the data management services themselves. The VOs may have their own GDH generation and validation schemes but in order to be certain, those schemes should be pluggable and complementary to the generic GDH validation scheme of the Data Grid. In order to scale well, the GDH validation and generation functionality should not be assured by a central service. Optimally it should even work on disconnected Grid nodes, i.e. a local service should be able to decide whether a new GDH is indeed unique or not.

Data location based on GDH or metadata: The Data Registry is the service that provides the mapping functionality between GDH and GDR. It should be a distributed service in the spirit of [26] that enables the location of local data even if the other Grid sites are unreachable. In the case of the location of data based on metadata, there will be higher-level services, probably databases and not just registries, that can execute complex queries to find data [24]. It is also necessary to be able to do the reverse mapping from GDR to GDH. There is no reason why a single GDR might not be referenced by more than one GDH.

Pre- and post-processing before and after data transfer: In addition to data validation, there might be necessary pre- and post-processing steps to be executed before and after data transfer respectively. For example, the data might need to be extracted, transformed into a format that can be transmitted, stripped of confidential data, etc before transfer. After transfer there may be additional steps necessary to store the data in the correct format in its designated store.

Replica Management: The replication process including processing, transfer, registration and validation should appear as a single atomic operation to the user. In order to assure correct execution of all steps, a replica manager service is necessary that orchestrates and logs each step and can take corrective action in the case of failures. It also needs to query either the destination data store to determine whether the application or user initiating the replication has the required access rights to the resource, and whether enough storage space is available. All replication requests should be addressed to the replica manager service.

Replica Selection: The job scheduler will need to know about all replicas of a given GDH in order to make a qualified choice as to where to schedule jobs, or whether replication or materialization of data needs to be initiated. The replica selection functionality of the data management services can select the optimal replica for this purpose.

Subscription to data: Automatic replication may be initiated by a simple subscription mechanism. This is very useful in VOs where there is a single data source but many analysts all around the world. Data appearing at the data source adhering to certain criteria can be automatically replicated to remote sites.

Replica Load Balancing: Based on access patterns of frequently used data this service can initiate replication to improve load balancing of the Data Grid. This service should be configurable so that many different strategies can be used to do automated replication [27].

Data Management Consistency: The data in all the metadata catalogs and the actual data store might get out of syncronization due to failures, or because the data was accessed outside the Grid context, etc. There could be a service enhancing this QoS aspect of data management that periodically checks the consistency of data in store with the data management service metadata.

Data Update: If data can be updated there are many different strategies regarding how to propagate the updates to the replicas. The strategy to be chosen is again dependent on the QoS requirements of the VO in question: how much latency is allowed, requirements on consistent state between replicas, etc. Of great interest are also the access patterns: are there many concurrent updates or only sporadic updates? The data management services need to be generic enough to accommodate many update strategies.

3.2.5 Metadata

Metadata can be classified into two groups according to their usage and content. Metadata associated with the operation of the system or any of its underlying components is designated “technical metadata”, whereas those metadata accumulated by the users of the system that pertain to the usage or classification of the corresponding data, is designated “application metadata”. Technical metadata is treated as metadata by the system and is essential to its successful operation. Application metadata are metadata managed by the Grid on behalf of the users and are treated simply as data by the Grid system. The boundary between these two groups is not always discernible nor unambiguous.

The GDR should contain most of the necessary technical metadata on the given physical instance of the data. This will be stored in the Data Registry. There is however a need for metadata services that store both technical and application metadata on all the other aspects of the data management services.

Security metadata: These include local authorization policies to be used by all data management services, specific authorization data for specialized high-level services. The details of the security infrastructure are not worked out yet, but these and other security metadata will certainly have to be stored in the Grid. See next subsection for more on security.

GDH metadata and data relations : As mentioned before, the GDH may serve as the primary key to other catalogs to add more metadata on the data elements.

Virtual Data Generation catalogs : The metadata stored in these catalogs should describe the programs to be invoked and their necessary parameters in order to materialize a virtual data set.

Replication progress data : As described above, the replica manager service acts on behalf of users or other services as an orchestrator of replication. It needs to control each step in the replication process and needs to make sure that failures are dealt with properly, hence the need for progress data. This may be stored in a dedicated metadata store or just log files, depending on the QoS requirements.

Subscription configurations : This is metadata to be used by the subscription service, defining the source and destination of the automated replication process. Also data on triggers like initiation time or quota usage level are stored here, etc.

Data access patterns : The automated load balancing mechanisms need to store monitoring data and other computed statistics in order to optimize data access.

Provenance data: A record of how data was created or derived may be generated automatically by the data grid technology, as well as by users or application code. Such data can be classed as both technocal and application metadata.

Durability and Lifetime data: Data indicating how long data should last, and whether its intermediate states should be recoverable.

Type and Format data: Tools used for the diagnostics and validation of data would benefit from metadata that describes the type and format of data.

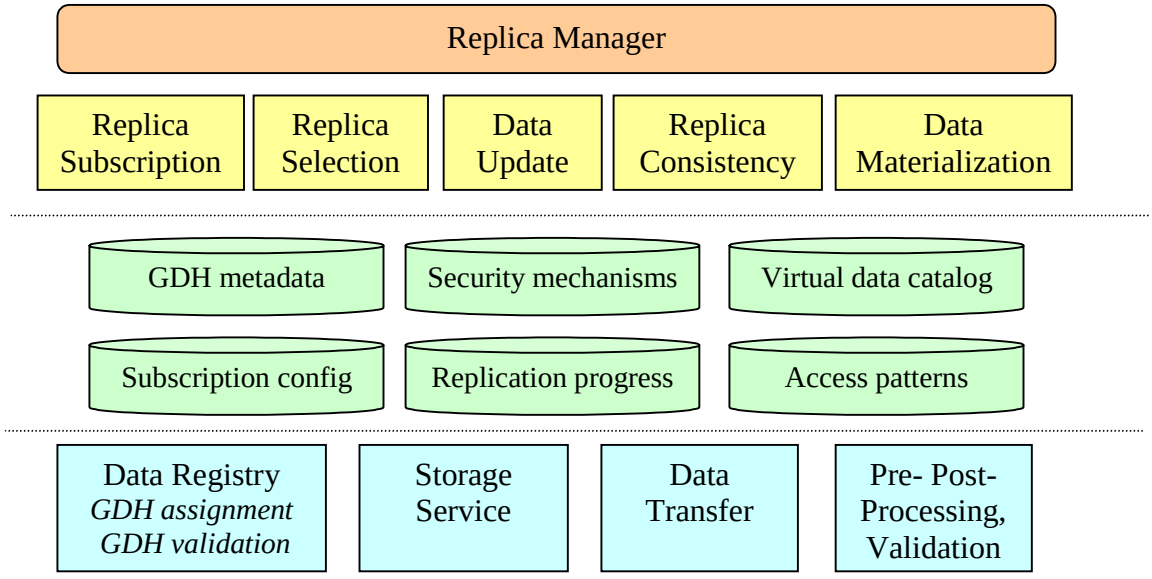

Figure 2: Elements of the data storage, transfer, management and metadata services and their logical layering. The top layer contains the high-level services with the replica manager being the 'chief orchestrator'. The middle layer shows the logical metadata services. The lowest layer contains the basic services.

3.2.6 Security

Data security is another one of the cornerstones of Data Grids. Certain industrial and research domains, such as biomedical research and the pharmaceutical industry, will impose very strict QoS requirements on security. Domains that deal with sensitive or proprietary data, as well as data the may lead to patents will probably develop their own security infrastructure to replace that developed by the Grid community.

In Grids the most basic functionality for security is authentication of users and groups using single-sign-on techniques and the possibility of delegating certain privileges to Grid Services to act on the user's behalf. In Data Grids there is a very strong emphasis on authorization, especially fine-grained authorization on data access and data update.

The resource providers also need to have an instrument by which they can enforce local policies – and are able to block malicious users locally without the need to contact a remote service. On the other hand, users need to be assured that their confidential data cannot be compromised by local site administrators. This calls for two-way authentication: the service needs to be authenticated with the user as well. As we will see later, security is a component that has received insufficient in the OGSA framework to date.

There are services that address security already such as the Community Authorization Service (CAS) by Globus [29] but they are not exploited yet in the Grid Services context.

3.2.7 Control and Monitoring

Most of the data management services need to be monitored and controlled by other services or users. The status of the services needs to be published, statistics need to be kept for optimization and accounting purposes. Services need to be deployed or terminated upon request, etc.

The services that we can identify here are service registries and factories, as proposed by the Open Grid Services Architecture. For gathering statistics there needs to be additional services for monitoring, one possibility is to set up services in the scheme of the Grid Monitoring Architecture that follows a consumer-producer model [12].

These services are essential for service and VO maintenance, as well as logging, accounting, billing and auditing. But also the security services will need a way to shut another service down if they detect that the service has been compromised. The OGSA mechanisms for lifetime management adequately address this issue.

3.2.8 Reliability and Fault Tolerance

In addition to control and monitoring services there need to be additional services that enhance the provided QoS with respect to reliability and fault tolerance. Upon occurrences of unpredictable failures, such as the sudden unavailability of resources, Grid nodes, networks, etc. dedicated services may choose to act in a predefined manner, failover to other sites or services for example.

In terms of fault tolerance, all services need to be implemented using robust protocols and mechanisms. OGSA does not enforce, but also does not hinder the usage of such protocols.

3.3 Data Grid and OGSA

In this section we investigate the steps that need to be taken to provide OGSA versions of the services described above. OGSA introduces several service concepts that need to be adopted in order to qualify as Grid Services. Necessary components are factories, registries and handle maps. Additional mechanisms that can be used to build our Data Grid are notification and lifetime management.

3.3.1 Factories

Factories have a very positive impact on the robustness and availability of services: If a service fails, it can be restarted by higher-level controlling services by calling on the service factory. Factories themselves may act as smart controlling services. This is also a very useful pattern for manageability, only certain factories need to be contacted to bootstrap or to disband a VO.

Transactions, which are very important in Data Grids, may be implemented easier in higher-level services by building upon the functionality of the factories. As an example, consider the Replica Manager service. The operation of replication involves data access checking, space reservation at the remote site, data transfer and lookup and update operations in many catalogs. To a high-level client these steps should be atomic: the Replica Manager needs to roll back upon failure or to retry on a different path if possible. The factory model facilitates the implementation of such transactional services considerably.

All of the services in Figure 2 could be instantiated through a set of factories, we could even have dedicated factories for each component. One attractive possibility is to build the replica manager with the factory interface to many of its fellow high-level services so that it can instantiate any of the services, providing the functionality upon request.

3.3.2 Registries

How should the registry interface be deployed in order to make service discovery possible? One trivial possibility is to implement the registry interface in all services so that they always have access to a registry to discover other services. The problem with such a model is scalability: if all services need to be notified of a service joining or leaving the Grid, the amount of messages will increase exponentially with the number of services. So it is much more sensible to keep just a few registries for a VO. They should be instantiated at Grid nodes that adhere to a higher QoS in terms of stability and accessibility of the node (for example it should be run at a large computing facility with good local support and not on somebody’s desktop machine in a university).

It also does not make sense to add the registry functionality to any other service because of the vital role of the registry. It should not be destroyed 'by accident' because the Grid service that hosts it was destroyed for some other reason. Nor

should there be services running that have additional functionality just because their registry functionality is still needed by other services – this opens up potential security holes. So registries are best deployed as dedicated services on high-quality Grid nodes.

3.3.3 Service Lifetime Management

Let's consider three simple models for service lifetime management.

Model 1: A very natural possibility is to keep a set of factories with a very long lifetime (possibly equal to the lifetime of the VO) that create the necessary services to assure the QoS requirements of the VO and keep them alive themselves using the OGSA lifetime extension mechanism. If a service fails, it should be restarted automatically.

Model 2: Another possibility is to set up factories that create new services upon application demand, in which case the applications would be responsible for keeping the services alive until they are no longer needed.

Model 3: A variant of model 2 is to not create a new service for each application, but to redirect incoming applications to existing services if they are not already overloaded and only to create new services for load balancing purposes.

The preferred model depends on the applications and the usage patterns of the VOs. If there are a lot of parallel applications that can easily share the same resource, model 1 or 3 is probably preferable – model 2 would generate too many services. If there are only a small number of very intensive applications, model 2 might be preferable. In the case of model 2 and 3 it is important that applications do not set very long lifetimes on the services since in the case of the failure of the applications, the services would remain too long.

Of course the different models may be mixed within a VO. The metadata services (middle layer in Figure 2) are less dynamic since they have persistent data associated with them, which is usually associated with a database or other data store. The VO would not want to let them expire if no application has used them for a while, hence model 1 is the most appropriate for these services.

The higher-level data management services as well as the lowest-level services may be created upon demand, they are much more light-weight, so model 2 and 3 might also be appropriate, depending on the VO.

3.3.4 Data Lifetime Management

Managing the data lifetime is orthogonal to managing the lifetime of the Data Grid services. Persistency requirements on certain kinds of data are so strong that their lifetime will most certainly exceed the lifetime of Grid service instances (identified by GSRs) – they depend on their hosting environment which evolve over time. Nevertheless, the GSH, the service that is associated with accessing persistent data need not change even if the implementation of the service has changed radically.

It is not up to the Grid to solve the issues of migrating data that has to be kept 'forever' to new storage technologies but it should provide an access point that does not change over time – the GDH. The GDR may also change radically, the schema of the data it points to may have changed but semantically it should still be the same data as it was 'forever before'. Information on migration of the data might be kept in the metadata catalogs as well if it is desirable.

3.3.5 A VO Example

What is the minimal set of services that a Grid node should have? It depends of course on what kind of resources the node site offers. If the site has only storage capabilities, it will need most of the data management services discussed above. If it only has computing capabilities but no storage capabilities, then it needs only a subset of those, but it will need job management services instead. A site offering both storage and computing capabilities will require both data and job management services.

The example shown in Figure 3 shows a simple VO layout with two sites and a possible set of services.

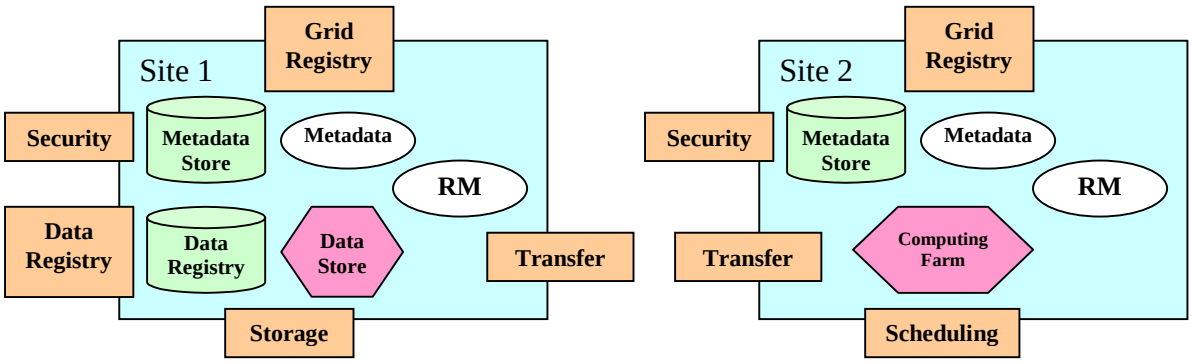

Figure 3: A simple VO providing services at two sites, one site capable of only storage and another capable of only computing. Ovals represent factories, rectangles represent exposed services. The Replica Manager (RM) factory may spawn other high-level data management services, see Figure 2

Of course there may be many other services such as monitoring and management services that are not shown in the picture.

4 Issues

It is understood that the OGSA work has just begun and there is a long way to go before we have a complete architectural specification in which all of the desired properties of Grids are addressed. This can only happen by having reference implementations and deployments of OGSA-compliant Grid middleware that will eventually expose the strengths and weaknesses of the architecture. OGSA will have to be refined and adjusted iteratively but this is a natural and healthy process and the first very important steps have been taken.

In this section shortcomings and open issues are discussed that indicate potential areas of future development. The current state of OGSA is analyzed by addressing each of the Grid properties we listed in section 2. We identify the areas that we feel are not adequately addressed by OGSA at this stage by referring to the tables in section 3.

4.1 Availability and Robustness

By introducing the Factory pattern, the availability and robustness of a service can be greatly improved in the context of data management services, as discussed in section 3. We have discussed how higher-level services might deal with failing or unavailable instances and start up new ones automatically.

Service data elements that deal with failing instances and policies on how to restart them need to be introduced. Possible items include policies that would answer the following questions: What happens if a service has been unavailable for a given time? How is service overload dealt with? What if the network of a VO becomes partitioned? What happens if the Factory or the Registry is suddenly unavailable? This also touches somewhat on the desired property of scalability.

4.2 Scalability

By designing OGSA to explicitly support transient services and dynamic service creation and by using soft state updates, scalability seems to be assured. In Table 1, scalability is dealt with in requirements 1 and 7, which in turn are addressed by several interfaces (see Table 2). We will have to see in the reference implementations and deployments of OGSA where the bottlenecks are and whether any architectural issues arise because of an eventual lack in scalability. As we have mentioned before, the robustness of the system needs to be ensured also in a scalable manner.

4.3 Monitorability

In the introduction of [2] it is mentioned that each VO needs certain levels of QoS to be achieved and that they may be measured in many different ways. The ability to set up VOs that fulfill many different QoS requirements is highlighted as one of the most desirable properties of Grids. OGSA does not elaborate further on QoS metrics: The Grid property MEASURABLE is not addressed by any OGSA requirement in Table 1. We agree that this area is a fundamental one to the success of Grids in general and needs to be dealt with in the very near future.

There needs to be not only agreed metrics of QoS, but defininitions from each Grid Service as to how it will enhance or decrease certain QoS metrics. We have outlined in the previous section how QoS might be measured for data management services, this needs to be done for all Grid Services in detail. There might be the need to define a QoS namespace to be able to query this property of services more explicitly in the WSDL description of the GSR. Each Service also needs to declare its own internal QoS metrics and give a value in a specific

instance in the case where different instances of the same service can be set up such that the given metrics can change.

Measurability is also very important when a VO defines its own set of necessary services or when it analyzes its own state and optimizes its performance due to changes in its nature or requirements. In order to define, bootstrap and optimize a VO it is essential to have QoS metrics by which the VO can measure itself and by which it can be measured by others, especially for billing purposes.

4.4 Integration

There is a need to integrate services not just within but also across VOs. OGSA solves this problem by defining the GridService interface and requiring all services to implement it. A lot of effort still needs to be put into the exact mechanisms and definition of common semantics in order that the integration of services (across VOs) may be achieved. In Data Grids, the integration of data access is also essential, hence our introduction of the concept of Grid Data References and the Data Registry.

4.5 Security

In the OGSA framework the hosting environment has the burden of authentication – which is reasonable – but there is no discussion on how local and VO-wide security policies are enforced. Is there the need for a Grid Service that deals with these issues or should each of the services have an interface addressing this, making it part of the GridService base interface? By deferring this problem to a later time, the design decision is made that security needs to be dealt with at the protocol layer or by the higher-level services.

By relying on Web Services, the strong industrial drive to come to a solution in this area will help speed up the process to design a suitable security infrastructure. Recent press releases by Microsoft and IBM have indicated the industry's commitment in this area. The release of WS-Security [31] however only deals with SOAP-based protocols, there is no effort to set up a security infrastructure spanning all possible protocols. There needs to be a lot of effort put into this domain also from the Grid community to check how existing Grid security infrastructures might interoperate with Web Service security mechanisms.

For Data Grids, authorization and accounting will be particularly complex for certain VOs. Experience shows that global authorization schemes don't work because local resources refuse to trust the global authorization authority to perform the authorization in accordance with the local policies – which may change over time. Also for logging, auditing and accounting purposes the local resource managers will always want to know exactly who has done what to their resource. An issue is how to delegate rights to automated Grid services that need to use the resources on behalf of the user even if the user did not initiate their usage explicitly.

Security will have to be dealt with very soon within OGSA since it will depend on the success of the underlying security framework. Open questions include: How are VO-wide policies applied? How are local security policies enforced? What is the role of hosting environments? How is an audit performed? Can a user belong to more than one VO and use both resources even if the security mechanisms differ?

4.6 Interoperability and Compatibility

Interoperability is explicitly mentioned as a requirement and is one of the driving concepts behind OGSA. Web Services, including Grid Services, are designed such that the modules are highly interoperable. There is no uniform protocol required that each service has to speak. WSDL descriptions are there to ensure interoperability.

The notification framework for passing messages between Grid Services is addressing this property explicitly. It resembles the concepts of the Grid Monitoring Architecture [12] but without the possibility to register notification sources with the target (sink) of the notification event. Interoperability is very closely related to discovery because services that need to interoperate have to discover among other things which common protocols they can use and whether there are issues of compatibility.

4.7 Service Discovery

As mentioned before, users of many services and services that want to interoperate need to get hold of the service descriptions to discover which services meet their needs or which services are still missing to achieve a given QoS. But how does the user know how to get hold of these descriptions? The OGSA's answer is the Registry and the HandleMap. The Registry needs to be searched to find the Grid Service Handles of the services that fulfill the user requirements – formulated in a query if necessary. The HandleMap then can be contacted to retrieve the detailed description of the services in question.

By holding a Grid Service Handle one can get at the corresponding (WSDL) description and the HandleMap is bound to the HTTP(S) protocol to assure availability of the description without another necessary discovery step.

This whole mechanism however leads us back to the Service Discovery problem: How do we get the GSH of the relevant registries in the first place? There has been significant effort in the P2P community to provide robust and scalable protocols addressing this issue, like Chord [36].

This touches again on the issue of QoS metrics and Service Data elements. How is it possible to get the handles of the registries that we can query to get a set of services that we might be interested in i.e. how do we find the registry or registries relevant for a given query? How is a query formulated to do so? OGSA considers using XQuery [10] to query the Service Data of the Registry, but then we need Service Data elements defining QoS. We tried to give a few examples for data management.

4.8 Manageability

Manageability is not dealt with explicitly in OGSA. The idea of unified monitoring and controlling mechanisms is there but not further exploited. We have discussed the manageability issues before with respect to VO management.

5 Summary

OGSA defines and standardizes a set of (mostly) orthogonal multi-purpose communication primitives that can be combined and customized by specific clients and services to yield powerful behavior. OGSA defines standard interfaces (portTypes in WSDL terminology) for basic Grid Services.

OGSA is an implementation independent specification. Several implementations of this specification may be offered. For example, a future Globus distribution will offer one or more reference implementations.

As with all complex systems, the devil is in the details. OGSA touches on the points of scalability, manageability and interoperability but there are many questions that remain unanswered or are deferred to a later time. How are QoS metrics defined and described? How does a VO get bootstrapped? How are VO-wide and local security policies enforced? How to deal with requirements of high availability and consistency? How do local resources – hosting environments – control and monitor their services and guard themselves against malicious abuse? It is understandable that OGSA is still in a very fluid state where lots of ideas are tested and evaluated before making it into the specification.

While being incomplete for the time being, OGSA provides a solid basis for future Grid infrastructures. The basic elements of OGSA have been analyzed and summarized in this article and we have tried to point out its current strengths and limitations.

We gave an overview on Data Grids and the kinds of data they manage and have tried to motivate many basic services from first principles: data transfer, storage, management and metadata services. We have introduced the notion of the Grid Data Handle that is the unique logical identifier of the data and the Grid Data Reference pointing to a physical instance and describing the data if necessary – in analogy to the Grid Service Handle and Grid Service Reference in OGSA. One of the real issues is how the data is actually located, we have introduced the notion of the Data Registry, which holds the GDH to GDR mappings. There are many issues with respect to data localization that we did not touch but which are dealt with elsewhere in the literature (see [4], [35], [35] and references therein).

We have tried to apply OGSA to Data Grids and have seen that all of the services can be deployed in the OGSA framework, which proves to be extensible and flexible enough to accommodate many different QoS requirements. All this is

a very preliminary, high-level view of Data Grids and OGSA which has to be refined and tested by actual implementations.

6 Acknowledgements

This work was inspired by many people, first and foremost by our friends and colleagues at CERN: Wolfgang Hoschek, Erwin Laure, Ben Segal, Heinz Stockinger, and Kurt Stockinger. We would also like to acknowledge Ann Chervenak, Ewa Deelman, Ian Foster, Carl Kesselman, Miron Livny, Arie Shoshani and Mike Wilde for many interesting discussions and insights.

Peter is most grateful to his family – Janka, Talia and Andor, for their incredible support.

7 References

[1] Ian, Foster, Carl Kesselman (editors) "The Grid: Blueprint for a new Computing Infrastructure", *Morgan Kaufmann Publishers 1999*.

[2] Ian Foster, Carl Kesselman, Jeffrey M. Nick and Steven Tuecke. "The Physiology of the Grid. An Open Grid Services Architecture for Distributed Systems Integration". *This volume*. http://www.globus.org/ogsa/

[3] Steven Tuecke, Karl Czajkowski, Ian Foster, Jeffrey Frey, Steve Graham and Carl Kesselman. "Grid Service Specification". Feb. 15 2002, http://www.globus.org/ogsa

[4] Wolfgang Hoschek. The Web Service Discovery Architecture. In Proc. of the Int'l. IEEE Supercomputing Conference (SC 2002) (to appear), Baltimore, USA, November 2002, see also his article in this volume.

[5] Dennis Gannon et.al. "The Open Grid Services Architecture and Web Services", this volume.

[6] Ian Foster, Carl Kesselman and Steven Tuecke. "The Anatomy of the Grid", *International Journal of Supercomputer Applications*, 15(3), 2001.

[7] E. Christensen, F. Curbera, G. Meredith and S. Weerawarana. "Web Services Description Language 1.1". *W3C Note 15*, 2001 http://www.w3.org/TR/wsdl

[8] World Wide Web Consortium. "Simple Object Access Protocol (SOAP) 1.1", *W3C Note 8*, 2000

[9] P. Brittenham, "An Overview of the Web Services Inspection Language", 2001, http://www.ibm.com/developerworks/webservices/library/ws-wsilover

[10] World Wide Web Consortium. XQuery 1.0: An XML Query Language, W3C Working Draft, December 2001.

[11] Wolfgang Hoschek, "A Unified Peer-to-Peer Database Framework for XQueries over Dynamic Distributed Content and its Application For

Scalable Service Discovery", PhD Thesis, Technical University of Vienna, Austria, 2002. http://edms.cern.ch/file/341826/1/phd2002-hoschek.pdf

[12] Brian Tierney, Ruth Aydt, Dan Gunter, Warren Smith, Valerie Taylor, Rich Wolski and Martin Swany. "A Grid Monitoring Architecture", *Grid Forum Working Draft GWD-Perf-16-2*, January 2002.

[13] 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", *J. Network and Computer Applications,* pp. 187-200, 2001.

[14] W. Allcock, J. Bester, J. Bresnahan, A. Chernevak, I. Foster, C. Kesselman, S. Meder, V. Nefedova, D. Quesnal, S. Tuecke; "Data Management and Transfer in High Performance Computational Grid Environments." *Parallel Computing*, 2002.

[15] H. Stockinger, A. Samar, B. Allcock, I. Foster, K. Holtman, B. Tierney. "File and Object Replication in Data Grids." *Proceedings of the Tenth International Symposium on High Performance Distributed Computing (HPDC-10),* IEEE Press, August 2001

[16] Arie Shoshani, Alex Sim, Junmin Gu, "Storage Resource Managers: Middleware Components for Grid Storage" *Proceedings of the 19th IEEE Symposium on Mass Storage Systems*, 2002

[17] Bob Jones (editor), "The EU DataGrid Architecture", *EDG Deliverable 12.4*, http://edms.cern.ch/document/333671

[18] Wolfgang Hoschek, Javier Jaen- Martinez, Peter Kunszt, Ben Segal, Heinz Stockinger, Kurt Stockinger, Brian Tierney, "Data Management (WP2) Architecture Report", *EDG Deliverable 2.2*, http://edms.cern.ch/document/332390

[19] A. Shoshani et.al, "SRM Joint Functional Design – Summary of Recommendations" *GGF4 Informational document Toronto 2002*, http://www.zib.de/ggf/data/Docs/GGF4-PA-Arie_Shoshani-SRM_Joint_func_design.doc

[20] The Griphyn Project Whitepaper, http://www.griphyn.org/documents/white_paper/white_paper4.html

[21] Ian Foster, Carl Kesselman, "A Data Grid Reference Architecture", *GRIPHYN-2001-12*, http://www.griphyn.org/documents/document_server/technical_reports.html

[22] Baru, C., Moore, R., Rajasekar, A. and Wan, M., "The SDSC Storage Resource Broker". In *Proc. CASCON'98 Conference*, 1998.

[23] Lee Luekig et.al, "The D0 Experiment Data Grid – SAM", *Proceedings of the Second International Workshop on Grid Computing GRID2001*, pp.177-184, Springer-Verlag 2001,

[24] Paul Watson, "Databases and the Grid" this volume.

[25] W. Allcock, J. Bester, J. Bresnahan, A. Chernevak, I. Foster, C. Kesselman, S. Meder, V. Nefedova, D. Quesnal, S. Tuecke "Data Management and Transfer in High Performance Computational Grid Environments". *Parallel Computing, 2002*

[26] A. Chervenak, I. Foster, A. Iamnitchi, C. Kesselman, W. Hoschek, P. Kunszt, M. Ripeanu, H. Stockinger, K. Stockinger, and B. Tierney, "Giggle: A Framework for Constructing Scalable Replica Location Services*," presented at Global Grid Forum, Toronto, Canada, 2001.*

[27] Mark Carman, Floriano Zini, Luciano Serafini, and Kurt Stockinger. "Towards an economy-based optimisation of file access and replication on a data grid". In *International Workshop on Agent based Cluster and Grid Computing at International Symposium on Cluster Computing and the Grid (CCGrid'2002).,* Berlin, Germany, May 2002. IEEE Computer Society Press

[28] P. Leach, R. Salz, "UUIDs and GUIDs", *Internet-Draft*, ftp://ftp.isi.edu/internet-drafts/draft-leach-uuids-guids-00.txt, February 24, 1997.

[29] L. Pearlman, V. Welch, I. Foster, C. Kesselman, S. Tuecke, "A Community Authorization Service for Group Collaboration", *Submitted to IEEE 3rd International Workshop on Policies for Distributed Systems and Networks,* 2001.

[30] EU DataGrid Data Management workpackage WP2. http://cern.ch/grid-data-management/

[31] B. Aitkinson et al, "Web Services Security (WS-Security)", Version 1.0 05 April 2002, http://www.verisign.com/wss/wss.pdf

[32] S. Abiteboul, P. Buneman and D. Suciu. "Data on the Web: From Relations to Semistructured Data and XML". *Morgan Kaufman, 1999*

[33] L. Guy, E. Laure, P. Kunszt, H. Stockinger and K. Stockinger: "Data Replication in Data Grids", *Informational document* submitted to GGF5, Replication-WG, http://cern.ch/grid-data-management/docs/ReplicaManager/ReptorPaper.pdf

[34] Karl Aberer, Manfred Hauswirth, Magdalena Punceva, Roman Schmidt, "Improving Data Access in P2P Systems", *IEEE Internet Computing*, 6(1), January/ February 2002.

[35] W. Hoschek, "Dynamic Timeouts and Neighbor Selection Queries in Peer-to-Peer Networks". In *Int'l Conf. on Networks, Parallel and Distributed Processing and Applications* (NPDPA 2002) (to appear), Tsukuba, Japan, October 2002.

[36] I. Stoica et al., "Chord:A Scalable Peer-to-Peer Lookup Service for Internet Applications", *Proc. ACM SIGCOMM*, ACM Press, New York,Aug. 2001, pp. 149-160.