Ann Chervenak, USC Information Sciences Institute
Karl Czajkowski, USC Information Sciences Institute
Mary Manohar, USC Information Sciences Institute

GGF DOCUMENT SUBMISSION CHECKLIST (include as front page of submission)	
	COMPLETED (X) - Date
1. Author name(s), institution(s), and contact information	(X) – February 23, 2004
2. Date (original and, where applicable, latest revision date)	(X) - February 23, 2004
3. Title, table of contents, clearly numbered sections	(X) - February 23, 2004
4. Security Considerations	(X) - February 23, 2004
5. GGF Copyright statement inserted	(X) - February 23, 2004
6. GGF Intellectual Property statement inserted.	(X) - February 23, 2004
7. **Document format : WORD**	(X) - February 23, 2004

GWD-I
Category: Informational

Ann Chervenak, USC Information Sciences Institute
Karl Czajkowski, USC Information Sciences Institute
Mary Manohar, USC Information Sciences Institute

OGSA Data Replication Services Working Group February 23, 2004

OGSA Replica Location Services

Status of This Memo

This memo provides information to the Grid community regarding the design of the Replica Location Service as an OGSI-compliant Grid service. It does not define any standards or technical recommendations. Distribution is unlimited.

Copyright Notice

Abstract

We describe issues relating to the design of an OGSI-compliant Grid service specification for the Replica Location Service (RLS). In particular, this design reflects recent discussions in the DAIS Working Group on the design of OGSA data services as well as the specification of ServiceGroups in the Open Grid Services Infrastructure version 1.0 specification. In version 2 of this specification, we discuss issues related to the emerging Web Service Resource Framework. In an appendix, we describe the implementation of an OGSI-compliant Replica Location Service.

Contents

Introduction

The OGSA Data Replication Services Working Group (OREP) is working toward the design of a Grid service specification for Replica Location Services. In this document, we describe our current thinking about RLS design. In particular, our design will be based on two specifications defined by other groups: the ServiceGroup primitive described in the Open Grid Services Infrastructure version 1.0 specification [4] and the data service specification being developed by the Data Access and Integration Services (DAIS) Working Group.

This document also presents a set of issues that will affect the RLS design with the emergence of the recently proposed Web Service Resource Framework (WS-RF) [5].

Finally, we present in an appendix the description of an implementation of the Replica Location Service design and its initial performance.

Aspects of a Grid service RLS include the following:

- Data services are uniquely identified by Grid Service Handles (GSHs)
- Replicated data sets are effectively members of an equivalence class according to some semantic definition of equivalence
- The replica set equivalence class should be exposed as a Grid service called a replicaSet service
- A replicaSet service design should be based on and extend the design of the ServiceGroup, which is a collection of Grid services.
- The replicaSet service should be have associated policies for authorization (who is allowed to add members to the replicaSet service) and semantics (what constitutes a member of the equivalence class)
- The RLS design may include additional indexes for aggregating information about multiple replicaSet ServiceGroups. These indexes should also be designed as extensions of ServiceGroups.

1. Background: Data Services and ServiceGroups

Our service-oriented Replica Location Service will depend heavily on two other aspects of grid services. We will be creating equivalence classes of data services that are defined to be replicas of one another. These data services are defined by the OGSA Data Service specification currently being developed through the GGF DAIS working group. The equivalence classes themselves, called replicaSet services, will be based on ServiceGroups as defined in the Open Grid Services Infrastructure (OGSI) specification, version 1.0. In this section, we briefly summarize the aspects of each specification most relevant to the design of a grid service-oriented Replica Location Service.

1.1 OGSA Data Services

The OGSA Data Services Specification [4] describes a data service as an OGSI Grid service that represents and encapsulates a data virtualization, which is an abstract view of some data. A data service has service data elements (SDEs) that describe key parameters of the data virtualization and operations that allow clients to inspect those SDEs, access the data, derive new data virtualizations, and manage data virtualizations. From the perspective of our discussion on Replica Location Services, the following aspects of Data Services are particularly relevant.

OGSI service data elements (SDEs) [2] are used to describe aspects of a data service's data virtualization as well as metadata about the virtualization.

OGSI Grid Service Handles are used to globally and uniquely identify data services.

Data services inherit basic lifetime management capabilities from OGSI Grid Services. Data services may be created dynamically using data factories.

Data services implement one or more of the four base data interfaces: DataDescription, DataAccess, DataFactory and DataManagement. For replica location services, we are most concerned with DataDescription, which defines service data describing the data virtualization and allows clients to inspect this service data. The DataDescription interface defines no operations, but the data service inherits data inspection (FindServiceData) operations from the Grid service portType and subscription/notification operations from the Notification portType.

The other data service interfaces are less likely to be used by the Replica Location Service. The DataAccess interface specifies SDEs and operations associated with accessing the data contained within the data virtualization. The DataFactory interface specifies SDEs that may be passed to a Factory CreateService operation to create a new data service derived from the existing data service. The DataManagement interface specifies SDEs and operations associated with configuring and monitoring the data service. While the DataAccess and DataFactory interfaces will likely not be used by replica location services, they would likely be needed by higher level data replication services that create new replicas from existing data sets.

1.2 OGSI Service Groups

In the Open Grid Services Infrastructure (OGSI) Version 1.0 specification, ServiceGroups are defined as Grid services that maintain information about a group of other Grid services [2]. The following aspects of ServiceGroups are particularly relevant to our discussion of Replica Location Services.

A ServiceGroup contains entries for member Grid services. Entries are represented as Service Data Elements (SDEs) of the ServiceGroup. Each ServiceGroup entry SDE value is a triple that contains the following:

- A locator called a serviceGroupEntryLocator
- A locator called a memberServiceLocator
- Content

In OGSI, a *locator* is a structure that may contain zero or more Grid Service Handles (GSHs), Grid Service References (GSRs) and/or Grid Service portType QNames [2], where the GSHs and GSRs refer to the same service instance.

The first locator specified in an entry SDE is the optional locator of the ServiceGroupEntry. The OGSI specification defines a ServiceGroupEntry portType as an optional interface through which individual entries in a ServiceGroup may be managed. A ServiceGroupEntry Grid service may be associated with a member Grid service of a ServiceGroup.

The second value in an entry SDE is the locator of the member Grid Service. The member locator identifies the service instance that is added to the ServiceGroup and that is described by the content of this entry.

The content value of the ServiceGroup entry SDE advertises some information about the member service instance. Based on the type of the member service, the ServiceGroup guarantees that certain content will be present according to membershipContentRules that can be associated with the ServiceGroup. Additional content may also be associated with the ServiceGroup entry.

Another optional portType associated with ServiceGroups is the ServiceGroupRegistration portType. This portType specifies the add operation, which creates a ServiceGroupEntry and adds it to the serviceGroup, and the remove operation, which removes one or more

ServiceGroupEntries from the ServiceGroup. This portType also provides the ability to specify extensibility declarations for add and remove operations. These extensibility parameters imply particular add and remove semantics.

The interface definition for ServiceGroups does not include any parameters related to specification of policies such as authorization.

2. Representing Replica Sets as Services

In our service-oriented approach to the Replica Location Service, we want to represent not only data items but also sets of replicas as Grid services. This will allow us to benefit from the same OGSI mechanisms for unique global naming, dynamic service creation, service data introspection and lifetime management that are provided for all Grid services.

In general, replicated data items are defined by an equivalence class. In a service-oriented RLS design, these equivalence sets would be exposed as services. Thus, we define a replicaSet Grid service as a virtualization of the set of replicas that make up an equivalence class. The equivalence class is globally and uniquely identified by a Grid Service Handle. Effectively, a replicaSet service provides a mapping from the handle or locator of the equivalence set service to one or more locators for member data services.

Information about data services that are members of the replicaSet service will be represented in service data elements (SDEs) of the replicaSet service. ReplicaSet service data may also include information about policies that the replicaSet service supports. A client may use standard inspection and subscription/notification methods to inspect a replicaSet service and obtain information about its members and policies.

A replicaSet services provide a convenient point for enforcing policies about the equivalence set of replicas. First, the replicaSet service can enforce access control policies about who is allowed to add new data services (or remove data services) as members of the equivalence set. The replicaSet service will only allow clients with whom it has an appropriate trust relationship to perform add or remove operations. Second, the replicaSet service can enforce semantic policies regarding the meaning of replication and which data service are allowed to be added to a replicaSet equivalence class. For example, the replicaSet could support policies that replicas can only be added to the equivalence class if the data service is an exact copy of the replicas in the equivalence class or is a version within a certain range of allowed versions of the data. The extent to which the assertions about replica semantics are verified or enforced depend on the replicaSet service implementation, as discussed in Sections 4 and 5. Finally, the replicaSet service can enforce policies about what attributes may be associated as service data elements (SDEs) of the replicaSet service.

The fact that replicaSets answer queries about their own members means that we do not require separate Replica Location Service index services from a functionality perspective. However, providing such indexes may be useful for availability and performance reasons. These indexes could aggregate information about data services that make up one or more replicaSet services. These indexes improve availability by answering queries about replicaSet members even if a particular replicaSet service itself is unavailable due to network partition or some other temporary failure. There might also be a performance advantage to aggregating replicaSet membership information and allowing bulk query operations on indexes. For example, an index might maintain a composed set of replicaSet descriptions, where each replicaSet description would specify the locator of the replicaSet service and a cached copy of the service data content made available by the replicaSet.

A scenario for creating a new replica and adding it to an existing replicaSet service might include the following steps:

- Client A invokes the data factory port type on an existing parent dataService to create a new derived data service that is a replica of the original service.
- Client A calls the add operation on the replicaSet service
    - This service will enforce authorization, semantic and possibly other policies to determine whether the client is allowed to create a new member of the equivalence set
    - If allowed, the new data service is added to the replicaSet service
- The replicaSet service may send information about its membership to one or more aggregating indexes

3. Using ServiceGroups to Design ReplicaSet Services

The design of the replicaSet service uses the ServiceGroup port types defined in the OGSI specification [2]. Since a ServiceGroup is a Grid service that maintains information about other Grid services, it provides a natural mechanism for grouping together information about data services that make up an equivalence set of replicas. In addition, there is a great deal of ongoing work in the OGSA community to refine and implement ServiceGroups as well as index servers that use ServiceGroups. We will take advantage of this ongoing development in designing replicaSet services.

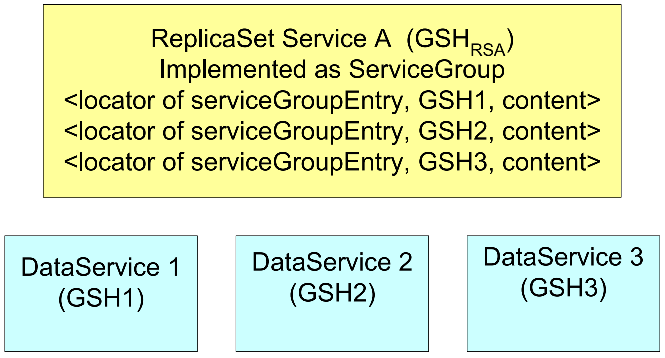
ReplicaSet Service A (GSH_RSA)
Implemented as ServiceGroup
<locator of serviceGroupEntry, GSH1, content>
<locator of serviceGroupEntry, GSH2, content>
<locator of serviceGroupEntry, GSH3, content>
DataService 1 (GSH1)
DataService 2 (GSH2)
DataService 3 (GSH3)

Figure 1: Shows a replicaSet service implemented as a ServiceGroup

Figure 1 illustrates a replicaSet implemented as a ServiceGroup. Each entry in the ServiceGroup is a service data element SDE consisting of three values: the locator of the serviceGroupEntry service used for management of the entry, the locator (in this case a Grid Service Handle) of the member data service, and content associated with the entry.

3.1 Creating replicaSet Services

The most basic required operation for replicaSet services is their creation using a replicaSetFactory. This factory would extend the ServiceGroupFactory service to support policy specification for authorization, replica semantics, etc. One possibility (discussed in more detail below) is for us to define some standard policies for replica semantics.

A replicaSetFactory service will contain SDEs with assertions that instances created by the factory can support. There may be mechanisms associated with these assertions. Different factory services may support different assertions, extensions and mechanisms. A call to a replicaSetFactory service to create a replicaSet service would include an extensibility element that should specify one of the advertised policies of the factory. The newly-created replicaSet service would include SDEs describing the policies that replicaSet supports; these SDEs could use the same schema to express these policies as the replicaSetFactory uses.

The replicaSetFactory relates to the Factory portType specified in the OGSI specification as well as the Agreement Factory being specified through the GRAAP Working Group of the GGF. If we eventually make replicaSet an agreement, then the policies can be published as part of the published agreement terms.

3.2 Service Data Elements and ServiceGroup Content

The replicaSet must have associated service data elements (SDEs) that describe the policies supported by the service, including authorization, replica semantics or other policies.

In addition, the replicaSet contains SDEs that describe its members. These SDEs are the entries already described containing the optional locator of the ServiceGroupEntry service, the locator of the member service, and content associated with the entry. Content in the SDE entries can be added at registration time or pulled later from the underlying data service.

One open question is what content needs to be associated with entries in the replicaSet ServiceGroup. One important component of the content field is an indication that the entry in the ServiceGroup represents a replica. This is needed because ServiceGroups are used for multiple purposes, and we want to distinguish replica entries from entries that might associate members by some other property than the replica equivalence measure. Clients should ignore any entries in the replicaSet that do not include required content fields.

Additional content will come from the individual data services that are members of the ServiceGroup. These member data services have service data elements (SDEs) that describe them. One option is to represent complete or partial member data service state in the content field of the corresponding replicaSet ServiceGroup entry. Another alternative is to summarize or aggregate data service SDEs in some manner for the content field of the replicaSet entries.

We would need to provide additional mechanisms to summarize or aggregate SDEs from a member data service. It is likely that these mechanisms would be useful for additional services besides ReplicaSets that use ServiceGroups.

We may find it useful to define a schema for the content entry in the replicaSet ServiceGroup. This schema could require that certain attributes must appear. In addition to the attributes defined by the underlying data services, we may define additional attributes specific to the replicaSet service.

3.3 ReplicaSet Service Methods

Because ReplicaSets will be based on ServiceGroups, they will inherit and extend the ServiceGroup PortType and possibly the ServiceGroupEntry and ServiceGroupRegistration PortTypes as well. The ServiceGroup portType publishes ServiceGroup entries as service data elements (SDEs), so these entries can be queried using standard GridService methods such as FindServiceDataByName and using the Notification portType to support subscription and notification. The ServiceGroupRegistration portType includes methods to add and remove entries from ServiceGroup. We would use the addExtensibility features of the ServiceGroupRegistration add call to define the content to be added to the replicaSet service entry, including which SDEs to take from the data service.

4. Enforcing Policy in ReplicaSets

As already noted, a replicaSet service can be an enforcement point for policies such as access control and replica semantics. The policy enforcement will be specific to a particular implementation of the replicaSet service. With respect to the interfaces for replicaSet services, there are two alternatives. We could have a variety of replicaSet services with different interfaces that reflect policies, for example, a byte-for-byte-copy replicaSet service or a versioning replicaSet service. Alternatively, we could create a generic replicaSet portType with different fields or parameters that may be used to specify replica policies.

Authorization policies provide protection for both replicaSets and clients. ReplicaSet services must trust the client that is attempting to add a member to the replicaSet service group to avoid malicious assertions about the membership of the replica equivalence class. Clients must trust the replicaSet service to make valid assertions about the members of the equivalence class.

Policies about the semantics of replication provide the ability to define equivalence classes as needed. These policy assertions are based on trust. Depending on the implementation of the replicaSet service, these assertions may or may not be enforced at the time when members are added to the equivalence class, and these assertions may or may not be maintained in response to changes in the content of data services that are members of the equivalence set. In the next section, we discuss replicaSet services that maintain these assertions despite updates to replicated data.

Some possible examples of standard semantic policies for replicas include the following:

- Byte-for-byte copy of data items, such as files
- Data objects that contain the same information in different formats
- Data objects that are equivalent to a specified degree
- Data objects that are derived from a common parent
- Versions of data objects
- Replicas that have been synchronized within a specific time period
- Partial replicas of data objects

The replicaSetFactory would include an interface defining policy, and each replicaSet instance would have SDEs that expose the policy supported by the instance. The policies supported by a replicaSet instance are not expected to change during the lifetime of the service instance.

5. Specialized ReplicaSet Services

We can implement specialized replicaSet services for implementing higher-level behaviors. For example, we can implement replicaSet services that maintain relationships among members of the equivalence class, such as byte-for-byte copy replication. We could use standard Grid service mechanisms such as subscription to be notified of any changes in the contents of data services and then propagate these changes among replicas according to a particular coherency scheme. Another option is to periodically introspect on the members of the replicaSet service to check coherence and remove non-complying members from the replicaSet equivalence class.

6. Issues for the Replica Location Service Design under WS-RF

The Web Service Resource Framework [5] is an emerging set of standards for specifying stateful resources in Web service environments. WS-RF standards are intended to eventually supercede the OGSI standards upon which the design of our Replica Location Service is based. These issues will need to be discussed at upcoming meetings of the OREP working group. Issues include:

1) Members of a replica set are likely to be stateful WS-Resources rather than OGSA data services or web services
2) ServiceGroups will still exist but will change. They will now group together WS-Resources and/or web services.
3) We will need to change our model for state associated with replicated data. Instead of having service data elements associated with a data service, we will now have resource properties associated with a WS-Resource
4) Instead of referring to data services using Grid Service Handles, we will now refer to WS-Resources via WS-Addresses.

7. Summary

We have presented an initial OGSA Grid service design of a Replica Location Service that groups together data services into equivalence classes called replicaSet services. The design of replicaSet services is based on OGSI ServiceGroups. We have discussed extensions that are needed to ServiceGroups to specify replicaSet service policies such as authorization and replica semantics. We have noted that enforcement of these policies will be provided by the implementation and will not have an impact on the interface specification.

Refinement of this initial design will occur through subsequent meetings of the GGF OREP Working Group.

8. Security Considerations

This discussion relates to OGSI-Compliant grid services for replica location. Therefore, our service will have all the same security capabilities and issues as other OGSI-compliant services. Additional security considerations such as access control over creation of replica mappings are discussed above.

9. Appendix: Implementation Experiences for the Replica Location Service

We have implemented a prototype ReplicaSet Grid service based on the OREP specification. In this appendix, we describe our implementation and present initial performance results. In particular, we describe the replica semantics and enforcement policies that our prototype supports, which are based on verifying the checksum of a file being added to a ReplicaSet.

9.1 The ReplicaSet Service Implementation

Our prototype implementation of a ReplicaSet service is instantiated in the Globus Toolkit version 3.0 Grid services environment. Because various aspects of Grid services and data services are still under development, in some cases our implementation simplifies functionality that will be more fully developed later.

9.1.1 Identifying Replicated Data Items

The Data Services specified by the OGSA Data Services Specification have not yet been implemented in the GT3 environment. For this reason, our implementation uses file URLs as locators for data objects. When data services are eventually implemented, these file URLs will be replaced by the Grid Service Handles of data services.

9.1.2 Replica Semantics and Policies

As already discussed, a range of replica semantics and enforcement policies can be supported in a ReplicaSet implementation. Replica semantics may require that data objects that are members of a ReplicaSet service are byte-for-byte copies of one another or versions of the same file. Enforcement policies range from no enforcement of replica semantics to enforcement of semantics at the time a member is added to the ReplicaSet to continuous enforcement of replica consistency.

Our prototype implementation currently supports two sets of policies for enforcing replica semantics. In our first set of policies, we assume a high level of trust between clients and ReplicaSet services and perform no verification at the time a member is added to a ReplicaSet. In other words, as long as a client is allowed by the authorization policies of the service (described below) to add a member to the ReplicaSet ServiceGroup, we perform no additional checks to verify that the member being added is actually a replica of existing members of the ReplicaSet by some semantic definition of replication.

We implement a second set of policies for enforcing replica semantics that requires verification that a member being added to a ReplicaSet service has exactly the same checksum as the first member added to a ReplicaSet service. This enforcement is only performed at the time a replica is added to the ReplicaSet. If replicas are later updated, the resulting inconsistency among replicas will not be detected by our ReplicaSet service.

One of the consequences of our policy enforcement is that adding a member to a ReplicaSet requires us to calculate the checksum for the data file. This requires transferring the file to local storage, performing the checksum calculation, determining whether the checksum matches that required by the ReplicaSet service, and deleting the temporary copy of the file. The overhead of performing this verification is proportional to the size of the file and can be substantial for large files, as shown in our performance results.

An alternative policy enforcement implementation would avoid the overhead of copying files and calculating checksums by allowing a client that wants to add a file to a ReplicaSet to assert a checksum value for the file. To accept such a signed checksum, our ReplicaSet service would need to have a sufficient trust relationship with the client to believe that the checksum assertion is valid. We plan to implement an assertion-based RLS in the future.

In the absence of such trust relationships, our current implementation performs verification of the checksum before allowing a new member to be added to the ReplicaSet. The first replica that is added to an empty ReplicaSet service represents the master copy. Subsequent replicas must have the same checksum as the master copy to be added to the ServiceGroup. By policy, we make this check only once when a new replica is added to a replica set. An alternate policy might require the replica set to periodically check that the files in the set match one another. If the service determines that replicas are inconsistent, it could remove non-complying members from the replica set.

9.1.3 ReplicaSet Factory

An instance of the ReplicaSet service can be created using a ReplicaSetFactory, which is an extension of an OGSI Factory service.

9.1.4 Authorization Policies

The ReplicaSet factory imposes authorization restrictions on ReplicaSet instance creation. It uses a standard Globus grid-mapfile that identifies the users who are allowed to create instances of the ReplicaSet service. Only users whose DN (Distinguised Name) is contained in the grid-mapfile are allowed to create instances of the ReplicaSet service.

There are also authorization restrictions imposed by each ReplicaSet service that determine who is allowed to add or remove entries in the ReplicaSet service. These restrictions are enforced based on a Globus grid-mapfile. One limitation of the current implementation is that we use a single grid-mapfile for all ReplicaSet instances; it may be preferable to have a different grid-mapfile associated with each ReplicaSet instance or a group of instances.

9.1.5 Port Types and Operations

Since the ReplicaSet service extends the OGSI ServiceGroup, we implement the ServiceGroup and ServiceGroupRegistration PortTypes of the OGSI specification [2]. ServiceGroupRegistration provides a management interface for a ServiceGroup. This portType has two functions, add and remove, which are implemented by the ReplicaSet service.

When a new replica is added to the ReplicaSet service, our implementation first copies the data file from the remote location to the local host using the RFT (Reliable File Transfer) Service or the GridFTP data transport protocol. We verify the checksum for the new replica instance, add the file to the ReplicaSet and delete the local copy of the file. The first replica added to a ReplicaSet is the “master copy” whose checksum must match all subsequent replicas. While OGSI ServiceGroups allow a member Grid service to be included in a ServiceGroup multiple times, our implementation only allows a particular file to appear once in a ReplicaSet service.

The remove operation removes an entry from the ReplicaSet. If a remove operation unregisters the last member of the ReplicaSet, then a subsequent add operation creates a new master copy and resets the checksum for the ReplicaSet service.

9.1.6 File Transfer for Add Operations

We use the Reliable File Transfer (RFT) Service to copy the file to local memory [9]. The RFT service transfers byte streams reliably. RFT is built on the top of GridFTP data transport client libraries. RFT maintains persistent state about outstanding transfers and restarts partially completed transfers after failures of the source or destination of the transfer.

We also provide a ReplicaSet implementation that directly uses the GridFTP transport protocol rather than the RFT service.

Our implementation defines two configurable parameters for file transfer: the local directory where a file can be copied for the checksum calculation and a maximum file size to limit the amount of data copied to the local machine.

9.1.7 Checksum Calculation

After the file is transferred, we verify that its checksum matches that of the master copy for the ReplicaSet service. We calculate an MD5 checksum for the file using the implementation provided by the Java 2 Runtime Environment version 1.4.0.

9.1.8 Service Data Elements and Related Operations

We defined two Service Data Elements (SDEs) for introspection by clients of the ReplicaSet service: *numOfReplicas* and *Checksum*. NumOfReplicas indicates the number of members of the ReplicaSet. The checksum SDE provides the checksum for the master copy of a ReplicaSet service, allowing users to verify the checksum of their own replica before attempting to add the replica as a member of the ReplicaSet service.

Although these SDEs may be queried using the findServiceData function provided by a GridService, we implemented the following access functions for the Service Data Elements.

- GetListOfReplicas: returns an ArrayList containing the URLs of the file replicas belonging to the ReplicaSet
- GetChecksum: returns a string representing the checksum for the files in the ReplicaSet
- GetNumOfReplicas: returns an integer number of members of the ReplicaSet

Table 1: Performance When Adding a Replica to a ReplicaSet Service Using GridFTP for File Transfer

File Size	***Total Replica***	***GridFTP Copy***	***Checksum Calculation***	***Additional Overhead***

	Addition Time at Client (seconds)	***(seconds)***	***(seconds)***	***(seconds)***
100 MB	19.433	10.0186	3.326	6.0884
500 MB	73.67	51.0804	16.155	6.4346
1 GB	146.13	102.794	37.136	6.2
2 GB	291.54	206.7534	78.053	6.774
4 GB	679.874	442.572	226.536	10.766
10 GB	1443.06	1030.4	405.268	7.397

9.1.9 Fault Types

Our implementation defines several faults that are thrown by the add operation. The *FileTooLargeException* is thrown when the replica's size exceeds the configurable maximum file size. The *ReplicaAlreadyExistsException* occurs when the replica is already a part of the ReplicaSet. The *FileCopyException* occurs when the copy of the file from the remote location fails. There are several reasons a copy operation may fail, including lack of permission to read the file on a remote host or write a copy locally and failure of the RFT service.

9.2 Initial Performance of the ReplicaSet Service Prototype

In this section, we present initial performance results for our ReplicaSet service implementation. The ReplicaSet service was implemented for Globus Toolkit version 3.0.2. Two workstations on a local area network were used in running these measurements. The first workstation is a single processor 2.0 GHz Intel Pentium machine running Red Hat Linux version 7.3. This workstation runs the hosting environment in which the ReplicaSet service is deployed as well as our client program that attempts to add replicas to a ReplicaSet service. This workstation is also the destination of data transfer operations and the machine that calculates MD5 checksums to verify that a replica may be added to the ReplicaSet service. The second workstation is a single processor 2.26 GHz Intel Pentium machine that runs Red Hat Linux version 8.0 and contains the files that are added as members of the ReplicaSet.

As described above, one variation of our ReplicaSet service implementation performs data transfers using the Reliable File Transfer Service. When measuring the performance of this implementation, we found that RFT data transfer times were highly variable. We are investigating the cause of this variability and plan to present the results in the final version of this paper.

We also implemented a variation of our Grid service that uses the GridFTP data transport protocol to transfer files. In Table 1, we present performance results for this implementation. We measured the time required to add a replica to a ReplicaSet service for files ranging in size from 100 megabytes to 10 gigabytes. Each data point in the table shows a mean of five ReplicaSet addition operations; these numbers exhibited low variance.

The second column in the table shows total observed time at the client for a replica addition to a ReplicaSet service. The other columns show the breakdown of this total observed time, including the time for the GridFTP copy operation, the checksum calculation and other overhead. The GridFTP data transfer time is proportional to the size of the file. The checksum calculation is roughly proportional to file size, with larger files requiring longer calculation times. The rightmost column shows other overheads, including Grid service overheads and the time required to add a new member to the ReplicaSet ServiceGroup. These overheads are fairly constant and do not depend on file size.

A final variation of our ReplicaSet service implementation performed no checksum verification. Only authorization policies were enforced; a client was allowed to add a replica to the ReplicaSet if the grid-mapfile of the ReplicaSet permitted this operation. Because no data transfer or checksum verification was required in this case, the time to add a replica to a ReplicaSet service depended only on Grid service overheads and not on file size. The mean time to perform a ReplicaSet addition operation was 5.5 seconds in this implementation. This value is similar to the overheads shown in the right column of Table 1.

9.3 Future Work

We plan to implement additional variants of the ReplicaSet Grid Service that support a variety of replica semantics and enforcement policies, including the assertion-based system mentioned earlier. We will perform wide area performance studies and evaluate the scalability of the ReplicaSet service at higher loads. We also plan to investigate the high variability we observed in RFT transfer times. In addition, we will implement a higher-level aggregating index for ReplicaSet services. Finally, we plan to incorporate the WS-Agreement [6] specification into our design for policy specification and enforcement.

The ReplicaSet service will evolve to accommodate the WS-Resource Framework [5] that was recently proposed to incorporate OGSI ideas into Web Service standards. In the WS-RF, the data services that we have used as the basis of our ReplicaSet service design will become stateful WS-Resources. The ServiceGroup mechanism in WS-RF will be similar to the one we have been using in our implementation, providing a grouping of WS-Resources. Thus, we expect that the changes to our design will require some refactoring and renaming but will not require fundamental logical changes to our model of replica location services.

9.4 Summary

We have presented a Grid service implementation of the Replica Location Service design being standardized in the OREP Working Group of the Global Grid Forum. Our implementation supports a particular set of replication semantics and enforcement policies: either no enforcement of replica semantics or enforcement that new members of a ReplicaSet are verified at the time of addition to be files with the same checksum as the master copy associated with a ReplicaSet. We presented initial performance results for our implementation. In our future work, we will experiment with a variety of additional semantic and enforcement policies and further evaluate the performance of our implementation.

Author Information

Ann L. Chervenak, USC Information Sciences Institute, 4676 Admiralty Way, Suite 1001, Marina del Rey, CA 90292, USA, annc@isi.edu

Karl Czajkowski, USC Information Sciences Institute, 4676 Admiralty Way, Suite 1001, Marina del Rey, CA 90292, USA, karlcz@isi.edu

Mary Manohar, USC Information Sciences Institute, 4676 Admiralty Way, Suite 1001, Marina del Rey, CA 90292, USA, annc@isi.edu

Intellectual Property Statement

References

[1] "Giggle: A Framework for Constructing Scalable Replica Location Services", Ann Chervenak, Ewa Deelman, Ian Foster, Leanne Guy, Wolfgang Hoschek, Adriana Iamnitchi, Carl Kesselman, Peter Kunszt, Matei Ripenu, Bob Schwartzkopf, Heinz Stocking, Kurt Stockinger, Brian Tierney to appear in Proceedings of SC2002 Conference, November 2002.

[2] "Open Grid Services Infrastructuure (OGSI) Version 1.0", S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, C. Kesselman, T. Maquire, T. Sandholm, D. Snelling, P. Vanderbilt, Global Grid Forum Open Grid Services Infrastructure (OGSI) Working Group, June 27, 2003.

[3] "Local Replica Catalog Service Specification (v2)", Ann Chervenak and Karl Czajkowski, Global Grid Forum OGSA Data Replication Services (OREP) Working Group, June 4, 2003.

[4] "OGSA Data Services", I. Foster, S. Tuecke, J. Unger, Global Grid Forum Data Access and Integration Services (DAIS) Working Group, August 14, 2003.

[5] "Modeling Stateful Resources with Web Services version 1.0", Ian Foster, et. al, http://www-106.ibm.com/developerworks/library/ws-resource/-ws-modelingresources.pdf

[6] "Agreement-based Grid Service Management (OGSI-Agreement) (Draft 0)", K. Czajkowski, A. Dan, J. Rofrano, S. Tuecke, and M. Xu.