

A Replica Location Grid Service Implementation

Mary Manohar, Ann Chervenak, Ben Clifford, Carl Kesselman

Information Sciences Institute, University of Southern California

Marina Del Rey, CA 90292

{mmanohar, annc, benc, carl}@isi.edu

Abstract

The OGSA Data Replication Services (OREP) Working Group of the Global Grid Forum has been working to develop standards for OGSI-compliant replica location services. This paper describes an implementation of the ReplicaSet service based on the OREP specification and evaluates the performance of different semantic enforcement policies.

1 Introduction

Scientific applications such as high-energy physics, climate modeling and computational genomics generate large volumes of data, measured in terabytes and petabytes. These large data collections are often stored in geographically distributed locations, and the communities using these data collections may also be large and geographically distributed. Performing computationally intensive analysis on these data collections requires efficient management and transfer of large volumes of data in wide-area environments. Often, this data management includes replication of data items to provide better performance via load balancing and higher availability by avoiding single points of failure.

A Replica Location Service (RLS) is one component of a Grid data management architecture [1]. An RLS is a fairly simple distributed registry that provides a mechanism for recording the existence of replicas and discovering them. An RLS service can be used to build higher-level data management services. For example, a Replica Management Service may be responsible for creating replicas, controlling replica access and maintaining consistency among replicas [7]. A Replica Selection Service chooses the best replica based on current resource or network conditions or based on criteria such as cost or security.

In our earlier work, we presented a framework for designing replica location services [1]. A Replica Location Service implementation is included in

Globus Toolkit version 3.0. The existing RLS implementation includes a multi-threaded RLS server that is not based on web service or Grid service technology.

The OGSA Data Replication Services (OREP) Working Group of the Global Grid Forum has been working to develop standards for OGSI-compliant replica location services. The latest version of this specification [3] discusses a ReplicaSet service design based on OGSI Service Groups [2] and OGSA Data Services [4][5].

We have implemented a prototype ReplicaSet Grid service based on the OREP specification. The ReplicaSet service is an OGSI-compliant Grid service that provides a mechanism to associate and discover replicas of a data item. The ReplicaSet is globally and uniquely identified by a Grid Service Handle (GSH). Effectively, a ReplicaSet provides a mapping from its handle to the handles of its members. Information about ReplicaSet members is represented as service data elements (SDEs) of the ReplicaSet service. A client may use standard inspection and subscription/notification methods to inspect a ReplicaSet service and obtain information about its members and policies.

In this paper, we describe our implementation and present initial performance results. In particular, we describe a variety of replica semantics and membership enforcement policies that can be supported by ReplicaSet services, and we discuss the particular policies that our prototype supports, which are based on verifying the checksum of a file being added to a ReplicaSet.

Although the ReplicaSet service is designed to be a component in an overall replica management system, its applicability is not limited to association of replicas. The extensions we define to OGSI ServiceGroups[2] can be generalized to provide mechanisms for creating associations among Grid services based on policies for authorization and membership maintenance.

2 Background: The ReplicaSet Service Design

In this section, we briefly describe the ReplicaSet service design that is being standardized through the OREP Working Group. This design depends heavily on two other aspects of Grid services: OGSA Data Services and OGSi ServiceGroups.

In a Grid services environment, data objects are OGSi-compliant *data services*. The OGSA Data Service specification [4] describes a data service as a 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. OGSi Grid Service Handles are used to globally and uniquely identify data services. Data services inherit basic lifetime management capabilities from OGSi Grid Services and may be created dynamically using data factories.

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]. A ServiceGroup contains entries for member Grid services. These entries are represented as Service Data Elements (SDEs) of the ServiceGroup. One value in an entry SDE is the locator (e.g., a Grid Service Handle) of the member Grid service. The ServiceGroupRegistration port type includes add and remove operations that are used to register and unregister entries from the ServiceGroup. Our design of a replica location Grid service extends the OGSi ServiceGroup, adding service data elements and modifying port type methods.

The ReplicaSet service that is being standardized through the OREP Working Group represents a set of replicated data objects as a Grid service. Important aspects of the ReplicaSet service design include:

- Replicated data items are OGSA Data Services that are uniquely identified by Grid Service Handles (GSHs).
- Data services form a replica set according to some semantic definition of replication.
- A replica set should be exposed as a Grid service called a ReplicaSet service.
- The ReplicaSet service design should be based on and extend OGSi ServiceGroups, which are collections of Grid services.

- The ReplicaSet should have associated policies for authorization and semantics.
- The RLS design may include additional indexes for aggregating information about multiple ReplicaSet ServiceGroups. These indexes should also be designed as extensions of ServiceGroups.

Figure 1 shows the ReplicaSet service implemented as an OGSi ServiceGroup, where the members of the ReplicaSet are OGSi-compliant data services. Each entry in the ReplicaSet service associates the Grid Service Handle of a member data service, a handle for managing the ReplicaSet entry and some optional additional content.

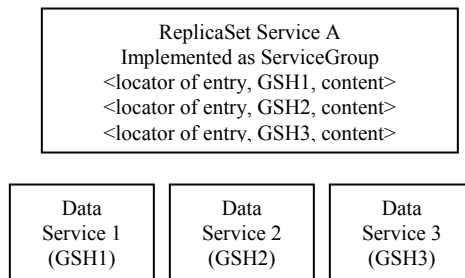


Figure 1: ReplicaSet Grid service and member data services.

A scenario for creating a new ReplicaSet and adding a member to the service would include the following steps. First, a client would request creation of a new ReplicaSet instance from a ReplicaSetFactory. If the client is authorized to create a new ReplicaSet instance, one will be created and the handle of the new service will be returned to the client. Next, the client will issue a request to add a data service as a member of the new ReplicaSet. The ReplicaSet will enforce authorization, replica semantics and other policies to determine whether the client is allowed to add a new member to the ReplicaSet.

2.1 ReplicaSet Policies

A key aspect of our ReplicaSet design is the ability to associate policies for authorization, replica semantics and semantic enforcement with a particular ReplicaSet service instance. The ReplicaSet provides a convenient point for policy enforcement.

ReplicaSet authorization policies specify who is allowed to create ReplicaSet instances or to add or remove ReplicaSet members. When a client attempts to perform one of these operations, the client's access rights will be checked, and any

unauthorized operations will be disallowed by the ReplicaSetFactory or the ReplicaSet service.

Semantic policies associated with a ReplicaSet define what constitutes a valid member of a replica set. Examples of replica semantics include exact byte-for-byte copies, versions of data items, partial replicas, replicas synchronized within a specified interval, or data items that contain the same content in different formats.

There may also be a range of policies for when and how replica semantic policies are enforced. At one extreme, a ReplicaSet could assume it is operating in a trusted environment and perform no verification that particular replica semantics are satisfied. At the other extreme, a ReplicaSet may continuously monitor its members to maintain consistency among them and remove non-complying replicas. One mechanism for continuous enforcement would be for the replicaSet service to subscribe to its member data services and receive notifications when updates occur to those member data services. Another enforcement scheme might verify that a new member joining a ReplicaSet service meets the semantic policy for replication at the time of addition, but may not monitor compliance thereafter.

The policies that a replicaSet service supports should be exposed as service data elements and should be discoverable through inspection or notification by clients accessing the replicaSet service.

In our prototype implementation, we experiment with the following policies: Grid Security Infrastructure (GSI) authentication; byte-for-byte copy replica semantics, which we verify by matching checksums of replicaSet members; and enforcement of checksum matching at the time a replica is added to the replicaSet.

3 The ReplicaSet Service Implementation

Next, we describe our prototype implementation of a ReplicaSet service for use 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.

3.1 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.

3.2 Replica Semantics and Policy Enforcement

As already discussed, a range of replica semantics and enforcement policies can be supported in a ReplicaSet. Our prototype implementation currently supports two sets of policies for enforcing replica semantics. In the first, 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 this enforcement scheme 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 the one 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. In this case, the ReplicaSet service would only need to verify that the asserted checksum matches the one required by the replicaSet. 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. We plan to experiment with consistency maintenance services in our future work.

3.3 ReplicaSetFactory and Authorization Policies

To create a new instance of a ReplicaSet service, a client makes a request to a ReplicaSetFactory, which is an extension of an OGSi Factory service. The ReplicaSet factory determines whether a client is allowed to create a new ReplicaSet instance by checking a standard Globus grid-mapfile. The Grid-mapfile contains the list of Distinguished Names (DNs) of those users who are allowed to create instances of the ReplicaSet service.

3.4 ReplicaSet Authorization Policies

Authorization restrictions are also enforced by each ReplicaSet service to determine whether a particular client is allowed to add or remove entries in the ReplicaSet service. This enforcement is based on a Globus grid-mapfile, which specifies the users allowed to add or remove members of a ReplicaSet.

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 with a group of instances.

3.5 Port Types and Methods

Since the ReplicaSet service extends the OGSi ServiceGroup, we implement the ServiceGroup and ServiceGroupRegistration port types of the OGSi specification [2]. The ServiceGroupRegistration port type provides a management interface for a ServiceGroup. This portType has two functions, add and remove, which are extended 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 that the checksum for the new replica instance matches the checksum required by the ReplicaSet, and if so, we add the file's URL 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 the checksum associated with the old master copy is deleted. A subsequent add operation to the ReplicaSet creates a new master copy and resets the checksum for the ReplicaSet service.

3.6 File Transfer

We use either the Reliable File Transfer (RFT) Service[6] or the GridFTP data transport protocol to copy a file to local memory before verifying the checksum and adding the file as a member of the ReplicaSet service. 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 or of the RFT service itself.

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 that limits the amount of data copied to the local machine.

3.7 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.

3.8 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 three access functions for the Service Data Elements. GetListOfReplicas returns a list of file URLs for members of the ReplicaSet; GetChecksum and GetNumOfReplicas return the corresponding SDE values.

3.9 Fault Types

Our implementation defines several faults that are thrown by the add operation. The *FileTooLargeException* occurs 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 as well as failure of the RFT service.

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

File Size	Total Replica Additon Time at Client (seconds)	GridFTP Copy (seconds)	Checksum Calculation (seconds)	Additional Overhead (seconds)
100 MB	19.322	9.747	3.023	6.552
500 MB	74.22	48.147	17.77	8.303
1 GB	143.39	97.89	38.14	7.36
2 GB	276.339	188.5	78.89	8.95
4 GB	543.146	376.37	158.01	8.76
10 GB	1351.38	936.78	406.45	8.148

Table 2: Performance Using Reliable File Transfer Service for File Transfer

File Size	Total Replica Additon Time at Client (seconds)	RFT Copy (seconds)	Checksum Calculation (seconds)	Additional Overhead (seconds)
100 MB	39.22	28.831	3.43	6.96
500 MB	110.89	86.9	16.68	7.31
1 GB	222.46	177.9	38.5	6.06
2 GB	435.22	343.23	83.19	8.8

4 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 dual processor 2.2 GHz Intel Pentium machine with hyper-threading enabled running Red Hat Linux version 9.0. 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 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 of 6.5 to 9 seconds are fairly constant and do not depend on file size.

Table 2 shows performance for our ReplicaSet implementation when using the Reliable File Transfer service to perform the data transfer before the checksum calculation. We are limited by the current RFT implementation to files of size 2 gigabytes or less. In each case, the time to add a replica in the ReplicaSet implementation that uses RFT is significantly longer than the time for the corresponding file size using the GridFTP implementation. While checksum calculation times and other overheads are similar for the two implementations, the data transfer takes significantly longer when using the Reliable File Transfer service. This is not only because invoking a grid service to perform the data transfer adds significant overhead, but also because RFT performs periodic checkpoint operations to save the state of the transfer to a database. This checkpointing provides the reliability of RFT, allowing the service to resume transfer operations after failures of source or destination machines or after the failure of the RFT service itself. For file sizes ranging from 500 megabytes to 2 gigabytes, the use of RFT increases the file transfer time by approximately 80% compared to transfers using GridFTP.

A final variation of our ReplicaSet service implementation included no verification or enforcement of the checksum. 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 approximately 5 seconds in this implementation. This value is similar to the overheads shown in the right columns of Tables 1 and 2.

The ReplicaSet enforcement policies that we have presented show significant performance differences, reflecting the tradeoffs that are inherent in defining policies for replica semantics and enforcement. In a highly trusted environment, overheads are low because no semantic verification is required. By contrast, an implementation that transfers files and verifies their checksums suffers a significant performance penalty, particularly for large files. It is notable that our implementation performs the checksum verification only when a file is added to a ReplicaSet; an enforcement policy that required continuous monitoring of replicas to ensure consistency would incur additional overheads.

5 Related Work

In this section, we describe how other systems manage replica location information.

Related Grid systems include the Storage Resource Broker [8] and GridFarm [9] projects that register, discover and maintain consistency among replicas using a metadata service. The design of the European DataGrid Project's Reptor system [7] includes a Replica Location Service implementation based on the RLS Framework [1]. The Reptor design is a Replica Management Service (RMS) that provides a single entry point for the user to access data in the RMS. Once a file is created or registered in the RMS, the system has complete control of the file. A client can access the file only through the RMS. Reptor is intended to manage replica consistency and also perform replica selection.

Next, we discuss replication and data location systems for peer-to-peer systems, including Chord, Freenet, Tapestry and OceanStore. Distributed peer-to-peer hash table systems such as Chord [10] and Freenet [11] perform file location by hashing logical identifiers into keys. Each node is responsible for a subset of the hashed keys and searches for a requested lookup key within its key space, passing the query to a neighbor node "near" in key-space if the key is not found locally [10]. Tapestry [12] nodes form a peer-to-peer overlay network. Each data object is associated with a Tapestry location root through a deterministic function, and this root is used for location purposes. OceanStore[13] employs a two-part data location mechanism that combines a quick, probabilistic

search with a slower, guaranteed traversal of a redundant fault-tolerant backing store.

The Grid service that is most similar to our ReplicaSet implementation is the Index Service provided in the Globus Toolkit Version 3 for the Monitoring and Discovery Service [14]. The Index Service provides an interface for accessing, aggregating, generating and querying service data associated with Grid Services. Like the ReplicaSet service, the Index Service's implementation is based on OGSi ServiceGroups. However, the Index Service is more generic in its functionality and does not perform any policy enforcement to validate entries added to the service.

6 Future Work

We plan to implement additional variations of the ReplicaSet Grid Service that support a variety of replica semantics and enforcement policies, including the assertion-based service mentioned earlier in which a trusted client is allowed to assert a checksum associated with a data item. This would improve service performance by removing the requirement of transferring the file to local storage and calculating the file checksum.

We will also conduct wide area performance studies and evaluate the scalability of the ReplicaSet service at higher request loads.

One aspect of the OREP specification that we have not yet implemented is the use of a higher-level aggregating index for ReplicaSet services[3]. Such an index would aggregate information about multiple ReplicaSet services and their member data services. Providing such indexes may be useful for availability and performance reasons. Indexes could improve availability by answering queries about replicaSets and their 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. We plan to implement such an index service in the future by extending OGSi ServiceGroups.

We also plan to incorporate the WS-Agreement [15] specification into our design for policy specification and enforcement. WS-Agreement will also likely be added to the OREP specification in the future.

The OREP specification and the ReplicaSet service design will evolve to accommodate the WS-Resource Framework [16] 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 likely become stateful WS-Resources. The ServiceGroup mechanism supported in WS-RF will be similar to the OGSi ServiceGroup that is the basis of our implementation and will provide 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.

7 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.

The extensions we have defined to OGSi ServiceGroups could be used to provide a general mechanism for grouping together Grid services that satisfy policy requirements.

8 Acknowledgments

This research was supported in part by the DOE Cooperative Agreements: DE-FC02-01ER25449 (SciDAC-DATA) and DE-FC02-01ER25453 (SciDAC-ESG).

9 References

- [1] A. Chervenak, E. Deelman, I. Foster, L. Guy, W. Hoschek, A. Iamnitchi, C. Kesselman, P. Kunst, M. Ripeanu, B. Schwartzkopf, H. Stockinger, K. Stockinger, B. Tierney, "Giggle: A Framework for Constructing Scalable Replica Location Services," SC2002, Baltimore, MD, 2002.
- [2] S. Tuecke, et. al, "Open Grid Services Infrastructure (OGSI) Version 1.0", Global Grid Forum Open Grid Services Infrastructure (OGSI) Working Group, June 27, 2003.
- [3] Ann Chervenak and Karl Czajkowski, "OGSA Replica Location Services", Global

- Grid Forum OGSA Data Replication Services (OREP) Working Group, September 19, 2003.
- [4] I. Foster, S. Tuecke, J. Unger, "OGSA Data Services", Global Grid Forum Data Access and Integration Services (DAIS) Working Group, August 14, 2003.
- [5] M. Antonioletti, M. Atkinson, S. Malaika, S. Laws, NW Paton, D. Pearson, G. Riccardi, "Grid Data Service Specification", DAIS-WG Informational Draft 9th Global Grid Forum, 14th August, 2003.
- [6] "The Reliable File Transfer Service", http://wwwunix.globus.org/toolkit/-reliable_transfer.html and <http://www-unix.mcs.anl.gov/~madduri/RFT.html>.
- [7] Guy, L., P. Kunszt, E. Laure, H. Stockinger, K. Stockinger, "Replica Management in Data Grids," Global Grid Forum 5, 2002.
- [8] Storage Resource Broker – Managing Distributed Data in a Grid Arcot Rajasekar, Michael Wan, Reagan Moore, Wayne Schroeder, George Kremenek, Arun Jagatheesan, Charles Cowart, Bing Zhu, Sheau-Yen Chen, Roman Olschanowsky, submitted to *Computer Society of India Journal, special issue on SAN*, 2003.
- [9] Tatebe, O., et al., *Replica Management of Grid Datafarm*: Global Grid Forum, Data Replication Research Group (GGF5).
- [10] I. Stoica, et. al, "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications," SIGCOMM Conference, 2001
- [11] I. Clarke, T. W. Hong, S. G. Miller, O. Sandberg, B. Wiley, "Protecting Free Expression Online with Freenet," *IEEE Internet Computing*, Vol. 6, No. 1, pp. 40-49, 2002.
- [12] Ben Y. Zhao, et. al. "Tapestry: A Resilient Global-scale Overlay for Service Deployment," *IEEE Journal on Selected Areas in Communications*, Vol 22, No. 1, January 2004.
- [13] John Kubiawicz, et. al, "OceanStore: An Architecture for Global-Scale Persistent Storage," Proceedings of the Ninth International Conference on Architectural Support for Programming Languages and Operating Systems (*ASPLOS 2000*), November 2000.
- [14] "Information Services in the Globus Toolkit 3.0 Release" <http://www.globus.org/mds>
- [15] K. Czajkowski, A. Dan, J. Rofrano, S. Tuecke, and M. Xu, "Agreement-based Grid Service Management (OGSI-Agreement) (Draft 0)".
- [16] Ian Foster, et. al, "Modeling Stateful Resources with Web Services version 1.0", <http://www-106.ibm.com/developerworks/library/ws-resource/-ws-modelingresources.pdf>
- [17] Globus Toolkit, www.globus.org
- [18] Borja Sotomajor, "The Globus Toolkit 3 Programmer's Tutorial."
- [19] GT3 API, <http://wwwunix.globus.org/-toolkit/-3.0/ogsa/impl/java/build/javadocs/>.
- [20] Palavalli, N., Shishir Bharathi, Ann Chervenak, Carl Kesselman, Robert Schwartzkopf (2004). Performance and Scalability of a Replica Location Service. Submitted to High Performance Distributed Computing (HPDC-13) Conference.
- [21] Czajkowski, K., S. Fitzgerald, I. Foster, C. Kesselman, "Grid Information Services for Distributed Resource Sharing," Tenth IEEE International Symposium on High-Performance Distributed Computing (HPDC-10), IEEE Press.