

# Combining Virtual Organization and Local Policies for Automated Configuration of Grid Services

Shishir Bharathi, Beom Kun Kim, Ann Chervenak, Robert Schuler  
*USC Information Sciences Institute*

**Abstract.** We investigate approaches for the automated configuration of distributed Grid services. In particular, we implement several approaches for combining configuration information specified by a group of collaborating institutions (a *Virtual Organization or VO*) with local configuration parameters. We describe our implementation of merging strategies for configuring the Globus Replica Location Service. Based on our initial work, we describe outstanding issues for merging local and VO configuration policies and for resolving conflicting policies.

**Keywords.** Distributed systems, Grid services, automatic configuration, policy management

## Introduction

Over time, Grids are increasing in size with respect to the number of resources and service instances being deployed, and these Grid resources and services are cooperating to perform increasingly complex tasks. Grid services provide well-defined functionality to clients, and their implementation may be either centralized or distributed. Distributed services provide better load balancing and higher tolerance to failure than centralized services.

An example of such a distributed Grid service is the Globus Replica Location Service (RLS) [1, 2], a distributed registry used for management and discovery of replicated data items. RLS is used by an increasing number of data-intensive Grid applications for management of replicated data. For example, the Laser Interferometer Gravitational Wave Observatory (LIGO) project [3, 4] deploys a distributed RLS at ten locations that contains registered mappings for over 40 million files. The Earth System Grid project [5] deploys RLS servers at four locations and stores mappings for over 400,000 files.

For distributed Grid resources and services to be effectively utilized, they need to be suitably configured and managed. This has proven to be challenging considering the number of services deployed, the needs of different applications, quality of service requirements, and the dynamic nature of grids. In addition, since all resources in a Grid do not fall under a single administrative domain, the restrictions imposed by each of the local system administrations must be respected in addition to any policies that are specified for the collaborating institutions.

Typically, the configuration of a Grid service or a resource deployed at a site has been the burden of the local system administrator. But, with larger deployments of

distributed Grid services and the need for coordination among multiple institutions, there is an increasing need for tools that provide flexible, scalable, automatic configuration of Grid services.

Earlier work has been done on automating configuration tasks. At one extreme is the scenario where all services fall under a single administrative domain (unlike a typical Grid), where automatic configuration is usually done with a centralized configuration manager that has global knowledge [6]. There might also be degrees of decentralization [7], but with the assumption that individual services or sites are not selfish and are interested mainly in the well being of the distributed system as a whole. At the other end of the spectrum, peer-to-peer file sharing systems [8-10] are composed of nodes that are individually administered. Each node is typically concerned with only its own well being without regard to the state of other nodes in the system. Grid services fall somewhere between these extremes.

In this paper, we first discuss the motivation for automated configuration of distributed Grid services. Next, we briefly describe the distributed Globus Replica Location Service and its configuration. Then we describe our initial approach to providing automated configuration of RLS servers and discuss alternatives for merging policy information specified by each local site and the collaborating institutions. Based on this initial work, we identify outstanding issues in merging conflicting configuration policies. We conclude the paper with a discussion of related work and our future plans.

## **1. Configuration of Distributed Grid Services**

A Grid is built when several collaborating institutions contribute resources to solve common problems [11]. Such a collaboration of personnel and resources is known as a Virtual Organization (VO). Each institution in the VO has its own set of policies that it would like to enforce on its resources and services, such as the access permissions granted to each user. At the same time, since it is part of the VO, the institution is also interested in cooperating with other sites and individuals in the VO. Further, the VO might have policies of its own that it would like to enforce on resources at the collaborating institutions.

Each of the policies that the VO or an individual institution wants to enforce produces a set of configuration settings for the services and resources in the VO. To determine how all these policies affect each other, resolve possible conflicts in these policies and produce the optimal configuration quickly is a complicated problem. Our research is concerned with identifying issues that arise when dealing with different sets of policies for configuring a distributed Grid service and providing automated mechanisms for determining optimal configurations.

In particular, we are interested in providing the capability for a Virtual Organization to specify most configuration policies for the services deployed at its collaborating institutions, but also allowing local administrators to override these policies as required. The advantage of allowing the VO to provide most of the configuration settings to each institution is that each local site configuration need not be modified every time the VO configuration changes. For example, consider the case when a new set of users is added to the VO. Without automatic configuration at the VO level, each configuration file at each site must be modified by local administrators to add access permissions for the new users. With automated configuration, a change at the VO level can be propagated and applied at each local site automatically.

One important consideration for automated configuration at the VO level is that local system administrators often require the ability to impose their own policies that may conflict with VO policies. For example, if a local site does not trust a particular user, it will deny that user access to its resources, even if that user is trusted by the VO and granted those access permissions at the VO level. The conflict between the desire to allow the VO to provide convenient, automated configuration at multiple institutions but still maintain local autonomy raises several challenging questions that are addressed later in the paper.

## 2. Background: The Replica Location Service

In this paper, we use the example of the Globus Replica Location Service (RLS) to highlight the problems of merging VO-specified and institution-specified configuration settings. Next, we briefly describe the RLS design and implementation, with particular emphasis on configuration of the distributed service.

The Replica Location Service is one component of a data management system for Grid environments. It is a registry that keeps track of where replicas exist on physical storage systems. Users or services register files in the RLS when the files are created. Later, users query RLS servers to find these replicas.

The RLS is a *distributed* registry, meaning that it may consist of multiple servers at different sites. By distributing the RLS registry, we are able to increase the overall scale of the system and store more mappings than would be possible in a single, centralized catalog. We also avoid creating a single point of failure in the Grid data management system. If desired, the RLS can also be deployed as a single, centralized server.

A *logical file name* is a unique identifier for the contents of a file. A *physical file name* is the location of a copy of the file on a storage system. The job of the RLS is to maintain associations, or *mappings*, between logical file names and one or more physical file names of replicas. The user can query an RLS server to find the logical file names, physical file locations, or attributes associated with them.

The RLS design consists of two types of servers: the Local Replica Catalog and the Replica Location Index. The *Local Replica Catalog (LRC)* stores mappings between logical names for data items and the physical locations of replicas of those items and answers queries about registered mappings. For a distributed RLS deployment, we also provide a higher-level *Replica Location Index (RLI)* server. Each RLI server collects information about the logical name mappings stored in one or more LRCs and answers queries about those mappings. The RLI index can be deployed in a hierarchical fashion to provide aggregation and redundancy of index information, as illustrated in Figure 1.

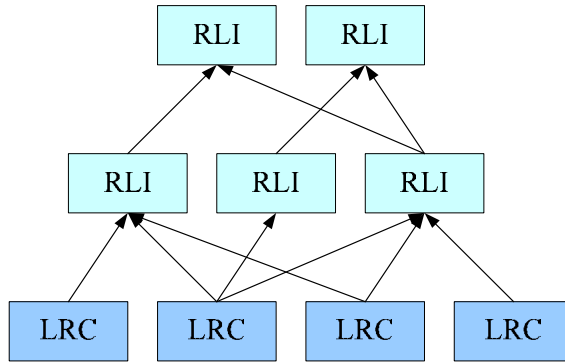


Figure 1: Example of a hierarchical, distributed RLS configuration

### 2.1. Configuring the RLS

The configuration settings for each RLS server are specified in a configuration file that resides on the same host as the service. These configuration parameters include access control lists that specify access permissions for users, the username and password used to access the back end database, timeout values for requests, and parameters that specify whether an RLS server acts as a local catalog, a distributed RLI index or both. In the remainder of this paper, we discuss different approaches for constructing these local RLS server configuration files by combining an overall VO configuration file with local configuration parameters.

## 3. Implementation and Policy Alternatives

We have implemented a prototype for performing automated configuration of a distributed RLS service. First, we set up an HTTP server for the Virtual Organization that contains the current configuration file to apply to all RLS servers in the VO. Then, we deployed a script that runs at each RLS site and performs automated configuration for the local RLS server. The script periodically polls the VO HTTP server to determine whether there have been any updates to the VO configuration file. If so, the script downloads the VO configuration policy file, merges that with a local configuration policy file using one of the schemes described below, and then applies the merged configuration to the RLS server at the site. Thus, there may be some delay between the time that changes are made on the VO server and when those updates are applied to all the RLS servers in the distributed deployment.

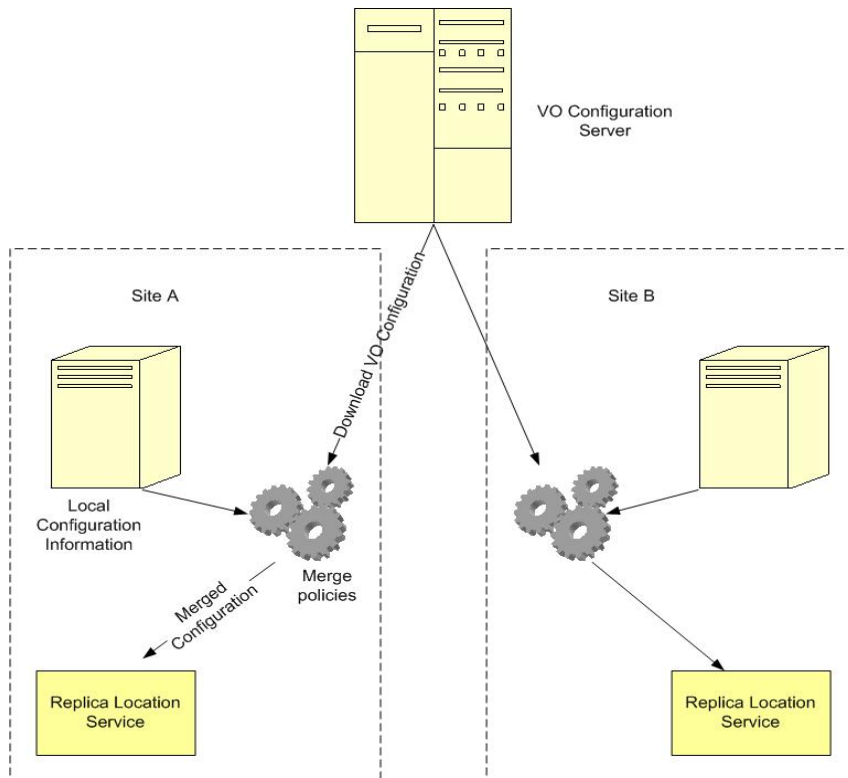


Figure 2 Merging of VO and local policies

Some RLS configuration parameters may be specified only in the local policy configuration, others only in the VO configuration, and some parameters may appear in both files before they are merged. For any parameter that appears in only one policy configuration file (local or VO), we apply the specified value to the final RLS configuration.

For some configuration parameters that appear in both local and VO configuration files, (e.g., the number of concurrent connections allowed, the number of threads servicing the requests, etc.), we let local policy override the VO policy. Some examples are shown in Table 1. The reason for this preference for local configuration parameters is that the local administrator is more likely to be aware of the limitations of the resources at that site and typically wants control over how local resources are used.

**Table 1:** Merging of policies

| <i>VO Policy</i>    | <i>Local Policy</i>   | <i>Final Policy</i>   |
|---------------------|-----------------------|-----------------------|
| max_connections: 20 | max_connections: 50   | max_connections: 50   |
| max_threads: 30     |                       | max_threads: 30       |
|                     | database_user: dbuser | database_user: dbuser |
|                     |                       | timeout: <default>    |

There are alternatives for handling these straightforward service configuration policies. For example, we could treat the policy value specified in the VO configuration

file as the minimum value for that parameter, and value specified in the local configuration file as the maximum value; the final configuration value applied to the RLS service could then be a value chosen from that range.

For other parameters, particularly those specifying access privileges for users, we need more sophisticated mechanisms for determining the final configuration values. Table 2 shows one scheme that we implemented for merging VO and local policies. Here the final set of privileges for the user is the *union* of the sets of privileges listed in the VO policy file and the local policy file. If a privilege appears in either the VO or the local configuration file, then it is added to the final configuration for the RLS server. A consequence of this union operation is that after the merge, the user might end up with more privileges than either the VO or the local site desired. In this example, user Alice ends up with lrc\_read privileges that the local site didn't want her to have and lrc\_write privileges that the VO didn't want her to have. Thus, a simple policy of merging configuration files with a union operation is unlikely to satisfy either the VO or the local site administrators.

**Table 2.** Union of user privileges

| <b><i>VO Policy</i></b> | <b><i>Local Policy</i></b> | <b><i>Final Policy</i></b> |
|-------------------------|----------------------------|----------------------------|
| Alice: lrc_read         | Alice: lrc_write           | Alice:lrc_read,lrc_write   |
| Bob: lrc_read           | Bob: lrc_read              | Bob: lrc_read              |
|                         | Carol: rli_read            | Carol: rli_read            |
| Dave:lrc_read,rli_read  | Dave: lrc_read             | Dave: lrc_read, rli_read   |

Table 3 shows another scheme that we implemented for merging policy files. Here, the final set of privileges for each user is the result of *intersecting* the privileges granted by the VO and the local site. This simple policy has the advantage that no access is granted to a user if it is denied by either the VO or the local site. In practice, however, this policy may be overly strict. For example, in Table 3 user Alice ends up with no privileges even though both the VO and local site wanted her to have certain privileges. The intersection merging scheme effectively requires that the local site must be involved in every authorization decision and must explicitly agree with any access rights granted by the VO before they can be applied at the RLS server. This eliminates one of the main advantages of doing automatic configuration, which is to allow a distributed service to be configured primarily based on VO policies without requiring the local site to be involved in every policy decision. For example, when a new user joins the VO and is added to the VO configuration file, the user will not be granted any access to local resources unless local configuration files are also updated to explicitly allow access for this user.

**Table 3.** Intersection of user privileges

| <b><i>VO Policy</i></b> | <b><i>Local Policy</i></b> | <b><i>Final Policy</i></b> |
|-------------------------|----------------------------|----------------------------|
| Alice: lrc_read         | Alice: lrc_write           | Alice:                     |
| Bob: lrc_read           | Bob: lrc_read              | Bob: lrc_read              |
|                         | Carol: rli_read            | Carol:                     |
| Dave:lrc_read, rli_read | Dave: lrc_read             | Dave: lrc_read             |

Our implementation and evaluation of these two schemes for merging local and VO configuration policies suggests that the union scheme is too permissive, while the intersection scheme is too restrictive. Next, we discuss outstanding issues related to merging these configuration files that will drive our future research.

Although we have used the examples of configuring Access Control Lists (ACLs) in this section, the issues involved in the merging of VO and local policies are just as relevant to other access control mechanisms. Role based access control, for example, introduces a level of indirection by mapping a user to a role and then assigning privileges to that role. Conflicts such as the ones described above arise when the VO and the local policies try to assign different roles to the same user or different privileges to the same role.

#### **4. Outstanding Issues**

We have noted the competing goals of providing the convenience of allowing most configuration management to be done automatically at the Virtual Organization level, while local sites still want to maintain autonomy and have the option of overriding VO policies. The desire for local site autonomy raises several interesting questions:

*How should VO and local policies be merged?* We have discussed several ways to merge VO and local policies. VO policies may take precedence over local policies or vice-versa, or we may choose one of the merging options (union or intersection or some other mechanism). Additionally, we might want some local policies to be merged with the VO policies, while other local policies should completely override the corresponding VO policies.

*To what extent should local policies dominate?* Allowing local policies to dominate might be desirable for some policy parameters. But should local access control policies be allowed to completely override the VO's policies? Some virtual organizations have expressed a desire for this functionality. However, considering that data stored in the local databases might belong to the VO and not to the individual institutions, it might not be advisable to allow local policies to override the access preferences of the VO. For example, in the case of the distributed Replica Location Service, if local policies allow access to the RLI index service to users that are not allowed those privileges by the VO, then the users may gain information about the location of data at other sites. This might not be what the VO or other sites want to allow.

There are additional security implications when a user is allowed to run a job at a site based on local access permissions despite a VO policy that tries to deny that access. This user job might need to access resources at other sites in the VO. The user's credentials have to be properly delegated to these other sites or else this will result in unauthorized operations.

*What are the implications for accounting systems of allowing local policies to override VO policies?* We must also consider other system management aspects such as accounting. If a local policy allows user Alice access to the VO resources even though the VO intended to deny that access, who should be charged for Alice's access? We will likely need to design accounting mechanisms that charge the VO when the policies are merged but charge the local site when local policies override VO policies.

## 5. Related Work

There has been considerable interest in the issues of dynamic resource configuration, authorization and access control and policy management. [12-16] We now briefly discuss some of the work in these areas that is related to the issues we have described.

Simple Network Management Protocol (SNMP) [6] based management systems are frequently used to manage routers, workstations and other kinds of network resources. The later versions of the protocol have addressed the shortcomings of the initial version by introducing decentralization, authentication etc. The Astrolabe management system [7] builds a hierarchy of resources and organizes them into zones. Agents running on each host communicate using Gossip protocols. Usually in such systems, all of the resources that are managed fall under the same administration and do not have to deal with the issues of conflicting policies. Yemini et al. [17] propose “management by delegation” which transfers the management responsibilities to the managed devices themselves in order to reduce the load on the manager. This approach requires that each resource be capable of self-management and reduces the burden on centralized managers. But, this in turn delegates the responsibility of policy conflict resolution to the individual resources.

Lupu and Sloman [18] describe the use of meta-policies to detect conflicts between policies that arise when using both negative and positive policies. They suggest that the conflict be resolved by giving a higher priority to negative policies. We could have potentially split up RLS ACLs into positive and negative ACLS and used a similar resolution. But, this would then prevent the local policy from allowing the user a certain privilege that the VO explicitly denies. Lupu and Sloman also briefly mention the conflicts that arise when multiple managers control the same resource. One open research question is whether detecting policy conflicts and passing them to a human administrator for resolution is sufficient or whether we can develop policies for resolving such conflicts automatically.

## 6. Future Plans

Many Grid services are now being designed according to the Web Service Resource Framework (WSRF) specifications [19, 20]. The WS-Resource Framework allows resources to export standard interfaces to query and modify resource properties. We expect to develop standardized configuration and management interfaces based on these interfaces. The framework also allows services and resources to *subscribe* to changes in the properties of other resources. The changes are propagated back to the subscriber as *notifications*. The subscriber can then react appropriately depending on the notification received. We expect this subscription/notification mechanism to be important in dealing with automatic configuration of Grid services.

The periodic polling operations performed by our current implementation could be replaced by a subscription/notification mechanism in WS-RF-based automated configuration management. Changes in the VO configuration would be propagated to the local sites faster and with less overhead. This approach would also make it easier to change resource configuration in response to dynamic events in the Grid. Standardized management interfaces based on WS-Resource Properties will allow for easier integration of new services into the VO management system.

While we expect the WS-Resource Framework to simplify the task of implementing automated management systems, the issues outlined in the previous sections still remain. We plan to continue discussions with the LIGO team as well as to discuss requirements with the Earth System Grid and TeraGrid projects and other virtual organizations. The large number of resources involved in the TeraGrid project and the sharing of resources by multiple VOs is expected to raise additional interesting issues for automated configuration.

We plan to explore the possibility of providing policy conflict information as feedback to other services in the grid. This information would be used to divert the flow of sensitive data away from the sites that have policy conflicts to those whose policies are a match with that of the VO's. In the case of projects like the Teragrid project, we expect policy conflicts to arise between different VOs sharing the same resources. We plan to investigate dynamic provisioning techniques to deploy and identify resources that match each VO's requirements.

## References

1. Chervenak, A., et al. *Giggle: A Framework for Constructing Scalable Replica Location Services*. in *SC2002 Conference*. 2002. Baltimore, MD.
2. Chervenak, A.L., et al. *Performance and Scalability of a Replica Location Service*. in *Thirteenth IEEE Int'l Symposium High Performance Distributed Computing (HPDC-13)*. 2004. Honolulu, HI.
3. Abramovici, A., W. Althouse, et al., *LIGO: The Laser Interferometer Gravitational-Wave Observatory*. *Science*, 1992. **256**: p. 325-333.
4. LIGO Project, *Lightweight Data Replicator*, <http://www.lsc-group.phys.uwm.edu/LDR/>. 2004.
5. Bernholdt, D., et al., *The Earth System Grid: Supporting the Next Generation of Climate Modeling Research*. *IEEE*, 2005. **93**(3): p. 485-495.
6. Case, J., Fedor, M., Schoffstall, M., and J. Davin, *A Simple Network Management Protocol (SNMP)*. Internet Request for Comments, 1990.
7. Renesse, R.v., K.P. Birman, and W. Vogels, *Astrolabe: A Robust and Scalable Technology for Distributed System Monitoring, Management, and Data Mining*. *ACM Transactions on Computer Systems*, 2003. **21**(2): p. 164-206.
8. Stoica, I., et al. *Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications*. in *ACM SIGCOMM*. 2001.
9. Ratnasamy, S., P. Francis, M. Handley, R. Karp, and S. Shenker. *A Scalable Content-Addressable Network*. in *ACM SIGCOMM*. 2001.
10. Chawathe, Y., et al. *Making Gnutella-like P2P Systems Scalable*. in *ACM SIGCOMM 2003*. 2003. Karlsruhe, Germany.
11. Foster, I., C. Kesselman, and S. Tuecke, *The Anatomy of the Grid: Enabling Scalable Virtual Organizations*. *International Journal of High Performance Computing Applications*, 2001. **15**(3): p. 200-222.
12. Poladian, V., Sousa, J.P., Garlan, D., Shaw, M. *Dynamic Configuration of Resource-Aware Services*. in *ICSE*. 2004.
13. Kon, F., et al. *Dynamic Resource Management and Automatic Configuration of Distributed Component Systems*. in *COOTS*. 2001.
14. Marshall, A.D., et al., *A Policy-Driven Approach to Availability and Performance Management in Distributed Systems*. 1997.
15. Lesser, V.R., *Cooperative Multiagent Systems: A Personal View of the State of the Art*. *IEEE Transactions on Knowledge and Data Engineering*, 1999. **11**(1): p. 133-142.
16. Oppenheimer, D.L., A. Ganapathi, and D.A. Patterson. *Why Do Internet Services Fail, and What Can Be Done About It?* in *USENIX Symposium on Internet Technologies and Systems*. 2003.
17. Goldszmidt, G., Yemini, S., Yemini, Y. *Network Management by Delegation - the MAD Approach*. in *CAS Conference*. 1991.
18. Lupu, E.C. and M. Sloman, *Conflicts in Policy-based Distributed Systems Management*. *IEEE Transactions on Software Engineering*, 1999. **25**(6): p. 852-869.
19. Czajkowski, K., et al., *The WS-Resource Framework Version 1.0*. 2004.
20. Foster, I., et al., *Modeling Stateful Resources with Web Services version 1.0*. 2004.