Integrating Grid Services using the DISCOVER Middleware[1]

Viraj N. Bhat and Manish Parashar
The Applied Software Systems Laboratory, Dept. of Electrical and Computer Engr.,
Rutgers, The State University of New Jersey, 94 Brett Road, Piscataway, NJ 08854
CAIP Technical Report-TR-268 {virajb, parashar}@caip.rutgers.edu

Abstract

Recent years have seen the development and deployment of a number of application/domain specific problem solving environments (PSEs) and collaboratories. These systems have evolved in parallel with the Grid and have been built on customized architectures and specialized technologies to meet unique user requirements and support specific user communities. While enabling these systems to share services and capabilities has many advantages, enabling such interoperability presents many challenges. In this paper we present the design, implementation and evaluation of the Grid-enabled Discover middleware substrate that enables Grid infrastructure services provided by the Globus Toolkit (security, information, resource management, storage) to interoperate with collaboratory services provided by Discover (collaborative application access, monitoring, and steering). Furthermore, it enables users to seamlessly access and integrate local and remote services to synthesize customized middleware configurations on demand.

1. Introduction

Grid computing [9] is rapidly emerging as the dominant paradigm of wide area distributed computing. Its goal is to realize a persistent, standards-based service infrastructure that enables coordinated sharing of autonomous and geographically distributed hardware, software, and information resources. The emergence of such Grid environments has made it possible to conceive a new generation of applications based on seamless aggregations, integrations and interactions of resources, services/components and data. These Grid applications will be built on a range of services including

[1] Support for this work was provided by the NSF via grants numbers ACI 9984357 (CAREERS), EIA 0103674 (NGS) and EIA-0120934 (ITR), DOE ASCI/ASAP (Caltech) via grant numbers PC295251 and 1052856.

multipurpose domain services for authentication, authorization, discovery, messaging, data input/output, and application/domain specific services such as application monitoring and steering, application adaptation, visualization, and collaboration.

Recent years have also seen the development and deployment of a number of application/domain specific problem solving environments (PSEs) and collaboratories (e.g. Upper Atmospheric Research Collaboratory (UARC) [19], Discover [16], Astrophysics Simulation Collaboratory (ASC) [24], NPACI HotPage [26], Environmental Molecular Sciences Collaboratory (ESML) [14], Diesel Combustion Collaboratory (DCC) [6], and Narrative-based, Immersive, Constructionist/Collaborative Environments for children (NICE) [23]). These systems provide specialized services to their user communities and/or address specific issues in wide area resource sharing and Grid computing. However, emerging Grid applications require combining these services in a seamless manner. For example, the execution of an application on the Grid requires security services to authenticate users and the application, information services for resource discovery, resource management services for resource allocation, data transfer services for staging, and scheduling services for application execution. Once the application is executing on the Grid, interaction, steering, and collaboration services allow geographically distributed users to collectively monitor and control the application allowing the application to be a true research or instructional modality. Once the application terminates data storage and clean up services come into play.

While enabling collaboratories/PSEs to share services and capabilities has many advantages, enabling such interoperability presents many challenges. The PSEs have evolved in parallel with the Grid computing effort and have been developed to meet unique requirements and support specific user communities. As a result, these systems have customized architectures and implementations, and build on specialized enabling technologies. Furthermore, there are organizational constraints that may prevent such interaction as it involves modifying existing software. A key challenge then, is the design and development of a robust and scalable middleware that addresses interoperability, and provides essential enabling services such as security and access control, discovery, and interaction and collaboration management. Such a middleware should provide loose coupling among systems to accommodate organizational constraints and an option to join or leave this interaction at any time. It should define a minimal set of interfaces and protocols to enable the PSEs to share resources, services, data and applications on the Grid while being able to maintain their architectures and implementations of choice. A key goal of the Global Grid Forum [13] and the Open Grid Services Architecture (OGSA) [12] is to address these challenges by defining community standards and protocols.

The primary objective of this paper is to investigate the design of a prototype middleware that will enable interoperability between PSE/collaboratory and Grid services to support the overall execution of computational applications on the Grid. In this paper we present the design, implementation and evaluation of the Grid-enabled Discover middleware substrate that enables Grid infrastructure services provided by the Globus Toolkit [10] to interoperate with collaboratory services provided by the Discover computational collaboratory, and enables users to seamlessly access and integrate local and remote services to synthesize customized middleware configurations on demand. This work builds on our previous work on the CORBA Community Grid (CoG) Kit [21] and Discover middleware [16].

2. The Grid-enabled Middleware Architecture

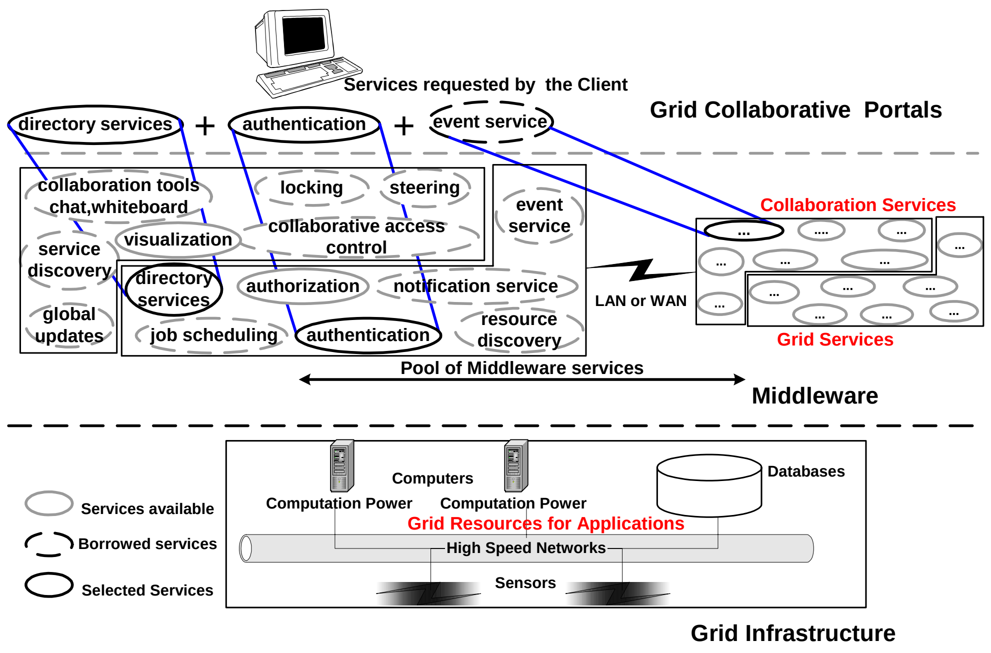


Figure 1: Discover Grid-enabled middleware for interoperable collaboratories.

The overall goal of the Grid-enabled Discover middleware substrate is to define interfaces and mechanisms for integration and interoperation of the services provided by Discover and the Globus Toolkit. A schematic overview of the middleware substrate is presented in Figure 1, and consists of a network of peer hosts that export a selection of services. The middleware essentially provides a "repository of services" view to clients and controlled access to local and remote services. It can be thought of as consisting of two service layers distributed across on the Grid – the *Grid Service Layer* and the *Collaboratory Service Layer* (see Figure 1). The collaboration service layer includes services for remote application access, collaborative application monitoring and steering, locking, and concurrency control. This layer builds on the Discover computational collaboratory, which consists of a peer-to-peer network of Discover interaction and collaboration servers and defines collaboratory services across these servers.

The Grid service layer includes infrastructure services such as resource discovery, authentication, security, directory services, resource management and scheduling. Some services, such as the event service, span both layers. The Grid services layer builds on the CORBACoG kit [21]. The CORBACoG provides access to CORBA server objects, which are wrappers around Globus Grid services. It also provides access to the CORBA Security Service and the CORBA Event Service. Note that all services in both service layers can be accessed by all clients (local and remote) connected to the middleware as long as they have appropriate access privileges – i.e. if certain services are not present at the local host they can be borrowed from a remote host. For example in Figure 1, the client uses locally available authentication and directory services and borrows the event service remote a remote host. The middleware combines these local and borrowed services and presents a virtual middleware to the client.

3. Implementation of the Grid-enabled Middleware Substrate

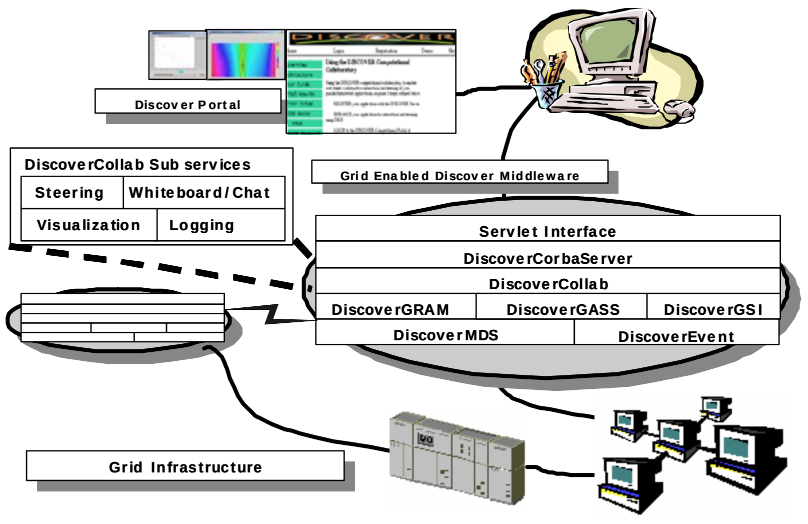


Figure 2: Conceptual architecture of the Discover Grid-enabled middleware.

An implementation overview of the Grid-enabled Discover middleware is presented in Figure 2. It consists of collaborative client portals at the front end, computational resources, services and applications at the backend and a network of peer hosts (servers) providing services in the middle. As mentioned above, the middle tier provides a repository of services view to the client and controlled access to Grid resources, services and applications. It also enables users to synthesize customized middleware configurations by combining local and remote services that they have access to. Clients are as simple as possible to ensure pervasive access. A client connects to its “closest” host and has access to all services based on its privileges and capabilities.

The prototype middleware substrate builds on CORBA/IIOP and provides peer-to-peer connectivity between hosts within and across domains. Server/service discovery mechanisms are built using the CORBA Naming [18] and Trader [27] services, which allows a server to locate remote servers and to access applications/services connected to the remote servers. Although CORBA does introduce some overheads, it provides sophisticated services such as security, discovery and naming and enables interoperability between servers. Furthermore, the use of IIOP can reduce client latencies when the communications are over large geographical distances as demonstrated in [16]. Note that XML based protocols (e.g. SOAP [25]) are popular technologies for service based distributed systems, the choice between CORBA IDL and XML in our prototype is a trade-off between speed and loose coupling. XML is self-describing and can provide a greater level of interoperability. However, XML parsing is still an overhead and is slower than CORBA IDL based object marshalling.

3.1. Discover Middleware Host (Server)

Discover interaction/collaboration servers build on commodity web servers, and extend their functionality (using Java Servlets) to provide specialized services for realtime application interaction and steering and for collaboration between client groups. Clients are Java applets and communicate with the server over HTTP using a series of HTTP *GET* and *POST* requests. Application-to-server communication either uses standard distributed object protocols such as CORBA or a more optimized, custom protocol over TCP sockets. An *ApplicationProxy* object is created for each active application/service at the server, and is given a unique identifier. This object encapsulates the entire context for the application. Three communication channels are established between a server and an application for application registration and updates, client interaction requests and application responses respectively. Core service handlers provided by each server include the *MasterHandler*, *CollaborationHandler*, *Command Handler*, *Security/Authentication Handler*, *Grid Service Handlers* (GSI, MDS, GRAM, GASS) and the *Daemon* servlet that listens for application connections. Details about the Discover Interaction and Collaboration servers can be found in [16] [17].

3.2. Discover Middleware Services

The Discover Grid enabled middleware substrate defines interfaces for three classes of services. The first is the *DiscoverCorbaServer* service interface, which can be generally termed as the service discovery service. This service inherits from the CORBA Trader service and allows hosts to locate services on demand. The second is the *DiscoverCollab* service interface, which provides uniform access to local or remote collaboratory services. Finally, the third class consists of interfaces to the Grid infrastructure services and provides uniform access to underlying Grid resources. This class includes the *DiscoverGSI, DiscoverMDS DiscoverGRAM, DiscoverGASS* and *DiscoverEvent* service interfaces. Each host that is a part of the middleware substrate instantiates CORBA objects that implement these interfaces and are essentially wrappers around the corresponding services. Each host implements the *DiscoverCorbaServer* interface and may implement one or more of the other interfaces.

***DiscoverCorbaSever*:** The *DiscoverCorbaServer* interface is implemented by each host and exports all available services at the host to the Discover middleware through the Trader service. Local services must register their presence with the *DiscoverCorbaServer* service to be discovered. A service description typically contains its name, location (i.e. address of its host) and its availability.

***DiscoverEvent*:** The *DiscoverEvent* interface is also implemented by each host. The *DiscoverEvent* service extends the CORBA Event Service [7] and enables users/services to monitor the status of applications and resources. The service defines an event channel at each host and clients/services can publish and subscribe to local and remote channels.

***DiscoverGSI*:** The *DiscoverGSI* interface represents the Globus GSI authorization and authentication service. It provides the basic security framework for the middleware substrate, and is used to create and delegate secure proxy objects on remote hosts and to enable secure access to local and remote (Collaboratory and Grid) services. *DiscoverGSI* uses Grid credentials provided by the user at login to delegate proxy objects.

***DiscoverMDS*:** The *DiscoverMDS* interface represents an instance of the Globus MDS service and provides access to information about Grid resources. The *DiscoverMDS* CORBA object accesses MDS information using the Java Naming and Directory Interfaces (JNDI) libraries. *DiscoverMDS* uses the *DiscoverEvent* service to publish updates to users and other services.

DiscoverGRAM**:** The *DiscoverGRAM* service represents the Globus GRAM service and allows clients to submit jobs on local and remote hosts. *DiscoverGRAM* objects works in coordination with the *DiscoverGSI* service for authorization and authentication with Grid resources. It also uses the *DiscoverEvent* service to receive updates regarding the status of jobs.

DiscoverGASS**:** The *DiscoverGASS* interface represents the Globus GASS service and enables users/services to access remote data and transfer data, application logs and applications executables. This enables applications to pre-stage data on remote machines, cache data, and log remote application outputs, and stage executables on remote computers. The *DiscoverGASS* service also allows clients to securely transfer files between source and destination pairs using the GridFTP [1] protocol, which also uses the *DiscoverGSI* service.

DiscoverCollab**:** The *DiscoverCollab* interface represents the collaboratory services provided by a host. This includes services for monitoring application status, application steering, locking and concurrency control, collaboration and visualization.

3.3. The Discover Portal

The Discover portal consists of a virtual desktop with local and shared areas. The shared areas implement a replicated shared workspace and enable collaboration among dynamically formed user groups. Locking mechanisms are used to maintain consistency. The base portal is presented to the user after authentication and access verification using Grid credentials. This provides the user with a list of available Grid and Collaboratory services that the user is authorized to access and allows the user to select the set of local or remote services to be used during the session. The application interaction desktop consists of (1) a list of interaction objects and their exported interaction interfaces (views and/or commands), (2) an information pane that displays global updates (current time step of a simulation) from the application, and (3) a status bar that displays the current mode of the application (computing, interacting) and the status of issued command/view requests. The list of interaction objects is once again customized to match the client's access privileges. Chat and whiteboard tools can be launched from the desktop to support collaboration. View requests generate separate (possibly shared) panes using the corresponding view plug-in. A snapshot of the Discover portal is shown in Appendix A.

4. Operation of the Discover Grid enabled Middleware

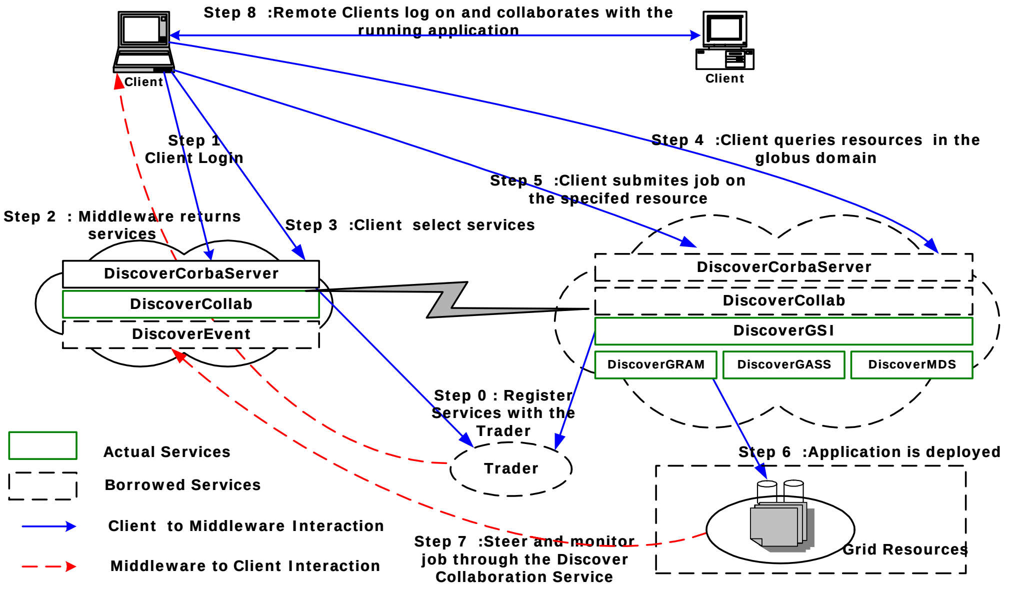

Step 8 :Remote Clients log on and collaborates with the running application
Client
Client
Step 1 Client Login
Step 4 :Client queries resources in the globus domain
Step 5 :Client submites job on the specifed resource
Step 2 : Middleware returns services
Step 3 :Client select services
DiscoverCorbaServer
DiscoverCollab
DiscoverEvent
DiscoverCorbaServer
DiscoverCollab
DiscoverGSI
DiscoverGRAM
DiscoverGASS
DiscoverMDS
Step 0 : Register Services with the Trader
Trader
Step 6 :Application is deployed
Actual Services
Borrowed Services
Client to Middleware Interaction
Middleware to Client Interaction
Step 7 :Steer and monitor job through the Discover Collaboration Service
Grid Resources

Figure 3: Operation of the Discover Grid-enabled middleware.

This overall operation of the Grid enabled middleware is illustrated in Figure 3. Each host joins the middleware and registers its services with the CORBA trader service (via the local *DiscoverCorbaSever* service). Each service is uniquely identified at the trader by its name and the machine address of its host. A client logging on to the middleware through the Discover portal first authenticates with the *DiscoverCollab* service. The client is then presented with a list of all services and applications, local and remote, to which the client has access privileges. The client can now interactively compose and configure its middleware stack using these services, and can use this customized stack and associated local and remote Grid as well as Collaboratory services to acquire resources, configure and launch applications, connect to, monitor and steer the applications, terminate applications and collaborate with other users. Note the client has to perform a second level of authentication with the *DiscoverGSI* service before accessing available resources, services or applications. The credentials presented by the client during this authentication are used to delegate the required client proxies. Through these proxies, clients can discover local and remote resources using the *DiscoverMDS* service, allocate resources and run applications using *DiscoverGRAM* service, monitor the status of applications and resources using the *DiscoverEvent* service and perform data/file transfer using the *DiscoverGASS* service. *DiscoverGRAM* also allows authorized users to terminate an application. The *DiscoverCollab* services enable the client to monitor, interact with and steer (local and remote) applications and to collaborate with other users connected to the middleware. Key operations are briefly described below.

Security/Authentication: The Discover security model is based on the Globus GSI protocol and builds on the CORBA Security Service. The GSI delegation model is used to create and delegate an intermediary object (the CORBA GSI Server Object) between the client and the service. The process consists of three steps: (1) Client and server objects mutually authenticate using the CORBA Security Service. (2) The client delegates the *DiscoverGSI* server object to create a proxy object that is authorized to communicate with other Grid Services. (3) The client can use this secure proxy object to securely invoke the services.

Each Discover server supports a two-level access control for collaboratory services: the first level manages access to the server while the second level manages access to a particular application. Applications are required to be registered with a server and to provide a list of users and their access privileges (e.g. read-only, read-write). This information is used to create customized access control lists.

Discovery of servers, applications and resources: Peer Discover servers locate each other using the CORBA trader services. The CORBA trader service maintains server references as *service-offer* pairs. All Discover servers are identified by the service-id "*Discover*". The service offer contains the CORBA object reference and a list of properties defined as name-value pairs. Thus the object can be identified based on the service it provides or its properties.

Applications are located using their globally unique identifiers, which are dynamically assigned by the Discover server and are a combination of the server's IP address and a local count at the server. Resources are discovered using the Globus MDS Grid information service, which is accessed via the *MDSHandler* servlet and the *DiscoverMDS* service interface.

Accessing Globus Grid services: Job submission and remote data access: Discover middleware allows users to launch applications on remote resources using the *DiscoverGRAM* service. Clients invoke the *GRAMHandler* servlet to submit jobs. The *DiscoverGRAM* service submits jobs to the Globus gatekeeper after authenticating using the *DiscoverGSI* service. The user can then monitor jobs using the *DiscoverEvent* service.

Similarly, clients can store and access remote data using the *DiscoverGASS* service. The *GASSHandler* servlet invokes the delegated *DiscoverGASS* service to transfer files using a client specified protocol.

Distributed collaboration: The Discover collaboratory enables multiple clients to collaboratively interact with and steer local and remote applications. The *Collaboration Handler* servlet at each middleware host handles the collaboration on its side, while a dedicated polling thread is used on the client side. All clients connected to an application instance form a collaboration group by default. However, as clients may connect to an application through a remote host, collaboration groups can span multiple hosts. In this case, the *DiscoverCollab* objects at the host polls other hosts for updates and responses.

The peer-to-peer middleware architecture offers two significant advantages for collaboration. First, it reduces the network traffic generated. This is because, instead of sending individual collaboration messages to all the clients connected through a remote middleware host, only one message is sent to that remote host, which then updates its locally connected clients. Since clients always interact through the host closest to them and the broadcast messages for collaboration are generated at this host, these messages don't have to travel large distances across the network. This reduces overall network traffic as well as client latencies, especially when the hosts are geographically far away. It also leads to better scalability in terms of the number of clients that can participate in a collaboration session without overloading a host, as the load now spans multiple hosts.

Distributed locking and logging for interactive steering and collaboration: Session management and concurrency control is based on capabilities granted by the middleware. A simple locking mechanism is used to ensure that the application remains in a consistent state during collaborative interactions. This ensures that only one client "drives" (issues commands) to the application at any time. In the distributed middleware case, locking information is only maintained at the application's middleware host i.e. the Discover middleware to which the application connects directly. The session archival handler maintains two types of logs. The first log maintains all interactions between a client and an application. For remote applications, the client logs are maintained at the middleware host where the clients are connected. The second log maintains all requests, responses, and status messages for each application throughout its execution. This log is maintained at the application's middleware host (the middleware to which the application is directly connected).

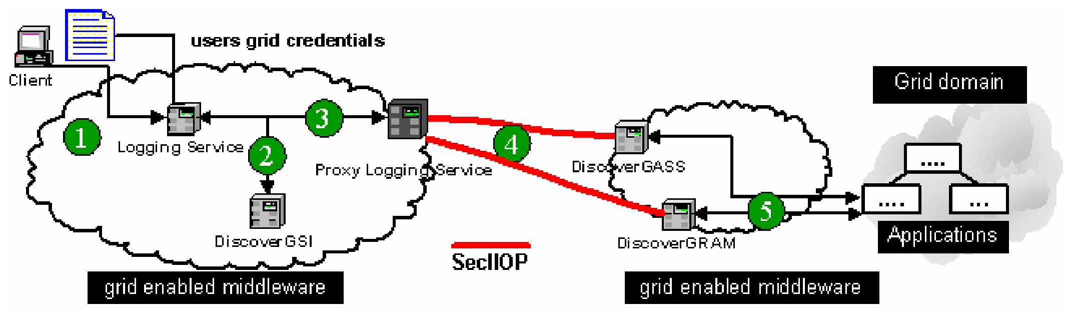


Figure 4: Delegation model across services.

As noted above the Discover Grid-enabled middleware enables local and remote services to be combined in an ad hoc way and collectively used to get achieved desired behaviors. For example, consider the scenario as illustrated in Figure 4. In this example, a client copies log files generated by the application during a run using a remote *DiscoverGASS* service. The client logs on to the middleware (step 1) and access the logging collaboratory service (part of *DiscoverCollab)*. The logging service uses the client's credentials and the *DiscoverGSI* service (step 2) to create and delegate a proxy logging service (step 3). This proxy logging services interacts with the *DiscoverGASS* service to transfer the log files to the local host (step 4). Note that these interactions are over a secure IIOP channel.

5. Experimental Evaluation

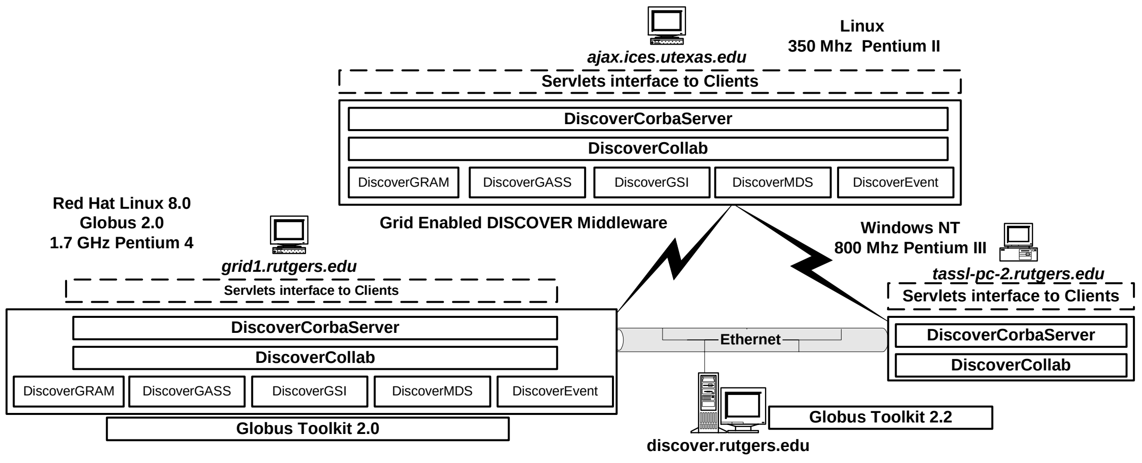


Figure 5: Experimental setup of the Grid-enabled Discover middleware.

The Grid-enabled Discover middleware is presently deployed at TASSL, Rutgers University and at the Center for Subsurface Modeling (CSM) and Institute for Geophysics (IG), University of Texas at Austin, and is used to enable multiple applications on the Grid from varied disciplines including reservoir engineering/subsurface modeling, seismic modeling, computational fluid dynamics, numerical relativity and astrophysics. We are currently expanding the network to include a deployment at University of Maryland and the Center for Advanced Computational Research (CARC), California Institute of Technology. The middleware implementation builds on commodity technologies including the Apache Tomcat Servlet engine and the JacORB [3] an open source implementation of the CORBA ORB.

This section presents an experimental evaluation of the Discover middleware. The overall setup for these experiments is show in Figure 5. It consisted of deployments at *grid1.rutgers.edu*, *discover.rutgers.edu* and *tassl-pc-2.rutgers.edu* at Rutgers University and *ajax.ices.utexas.edu* at University of Texas. Deployments at *grid1.rutgers.edu* and *ajax.ices.utexas.edu* had complete installations (Grid and Collaboratory services) while *discover.rutgers.edu* had only Grid services and *tassl-pc-2.rutgers.edu* had only Collaboratory services. We used the transport equation application kernel with adaptive mesh refinement (*tportamr)* for our experiments. The application was run on Beowulf clusters at Rutgers. The evaluations consisted of evaluating the latencies in accessing local and remote services over local and wide area networks and are presented below.

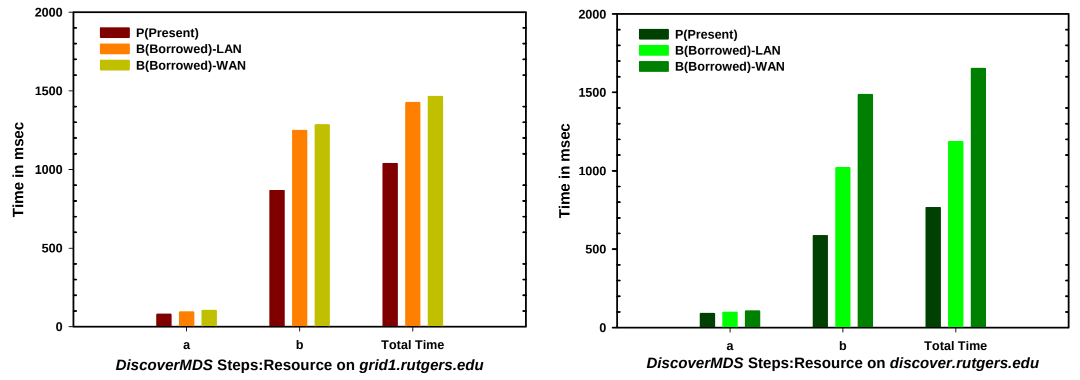
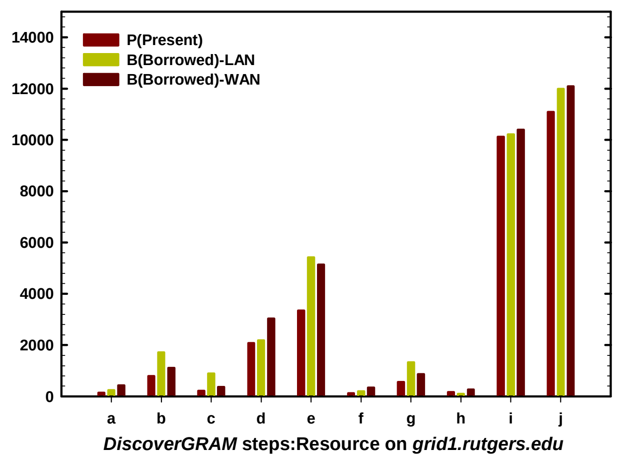


Figure 6a, 6b: *DiscoverMDS* service discovers resources on *grid1.rutgers.edu* (6a) and *discover.rutgers.edu* (6b). *a* represents the time to locate the *DiscoverMDS* service and *b* represents the time to query the resources on the selected host.

Evaluation of the *DiscoverMDS* service: The evaluation of the *DiscoverMDS* service is divided into three cases. In the first case the *DiscoverMDS* service is locally present (case P). In the second case the *DiscoverMDS* service is borrowed from a remote host over LAN (case B-LAN). In the third case the *DiscoverMDS* service is borrowed from a remote host over WAN (case B-WAN). In all three cases clients used the *DiscoverMDS* service to discover resources at Rutgers. In each case, the experiment consists of two steps: (a) discovering the *DiscoverMDS* service using the CORBA Trader service and (b) invoking the service to discover resources. The times for steps (a) and (b) for discovering resources on *grid1.rutgers.edu* and *discover.rutgers.edu* are plotted in Figure 6a and 6b respectively. As seen in the plots, the time for discovering the service (step a) is small compared to the time for querying for resources (step b). This is primarily because of the overheads of querying MDS and packing, transporting and unpacking the large amount of returned resource information. Note that the average time for querying resources on *discover.rutgers.edu* is larger than that for *grid1.rutgers.edu* as *discover.rutgers.edu* is a 16 node cluster while *grid1.rutgers.edu* is a single processor machine.



a	**Resolving services: *DiscoverGRAM***
b	**Delegation: *DiscoverGSI***
c	**Event channel creation: *DiscoverEvent***
d	**Job start time on *grid.1.rutgers.edu***
e	**Total time to start job – i.e. a+b+c+d**
f	**Resolving services: *DiscoverGRAM***
g	**Delegation with the *DiscoverGSI***
h	**Event channel creation: *DiscoverEvent***
i	**Job cancellation time: *DiscoverGRAM***
j	**Total time to cancel job – i.e. f+g+h+i**

Figure 7: *DiscoverGRAM* service launches the *tportamr* application using steps a, b, c, and d, and terminates the *tportamr* application using steps f, g, h, i on *grid1.rutgers.edu*.

Evaluation of the *DiscoverGRAM* service: The evaluation of *DiscoverGRAM* consisted of using the service to launch and terminate the *tportamr* application on *grid1.rutgers.edu*. Application deployment consisted of the following steps: (a) discovering the *DiscoverGRAM* service, (b) using *DiscoverGSI* to delegate a service proxy, (c) create an event channel for application monitoring, and (d) launch the application on the selected host i.e. *grid1.rutgers.edu*. Application termination similarly consisted of the following steps: (f) discovering the *DiscoverGRAM* service, (g) using *DiscoverGSI* to delegate a service proxy, (h) creating an event channel for application monitoring, and (i) terminate the application selected. Note that the resource for launching the application and the application to be terminated are discovered and selected using the *DiscoverMDS* service. The times required for each step are plotted in Figure 7.

As in the previous experiment, we consider three cases: in case P, the required services are local, in case B-LAN, the required services are borrowed over LAN, and in case B-WAN, the required services are borrowed over WAN. Note that the times for lauching and terminating the application are quite comparable for the three cases. The large termination time is due to the cleanup performed by GRAM.

Evaluation of the *DiscoverGASS* service: The evaluation of the *DiscoverGASS* service consisted of using the service to transfer files of different sizes. We measured the time required to transfer files between *grid1.rutgers.edu* and *discover.rutgers.edu*. In this experiment we considered the case P where the *DiscoverGASS* service was locally present. The measured transfer time and the file sizes in bytes are plotted in Figure 8 using a log-log scale. The file sizes and the transfer times varied exponentially and ranged from 2 bytes to approximately 10 MB and the corresponding transfers times varied from 9 msec. to 637 msec. respectively. It can be seen that the *DiscoverGASS* performed well for small and medium file sizes (9 msec. for ~2 bytes and 47 msec. for ~1 MB). However the performance rapidly deteriorated (637 msec.) as file sizes approached 10 MB. Note that the typical size of a log files generated

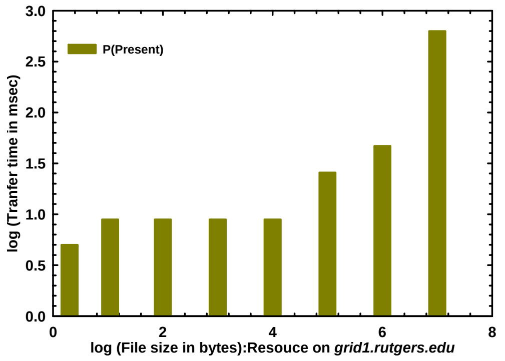


Figure 8: Log-Log plot of transfer times for various files sizes using *DiscoverGASS* service (P case).

during the *DiscoverGRAM* experiment was around 100 KB. We are currently evaluating cases where the service is borrowed over LAN (B-LAN) and over WAN (B-WAN).

Evaluation of the *DiscoverCollab* service: The evaluation for Collaboratory services (access latency over local area and wide area networks, effect of multiple clients on access latencies and server memory overheads due to local and remote applications) was presented in [16]. This evaluation consisted of measuring scalability, response times and latencies when multiple clients collaboratively interact with an application. These measurements were conducted for cases where the *DiscoverCollab* service is local, borrowed over a LAN and borrowed over a WAN. The results showed that although response times were larger when using borrowed services, the overhead was constant for large response sizes. Furthermore, when using the WAN, the results showed the benefits of the hybrid P2P design and the use of IIOP. The results also demonstrated that the middleware scaled to over 20 (distributed) collaborating clients simultaneously interacting with an application.

6. Conclusions

This paper presented the design, implementation, operation and evaluation of the Discover Grid-enabled middleware substrate. The middleware substrate enables Grid infrastructure services provided by the Globus Toolkit (security, information, resource management, storage) to interoperate with collaboratory services provided by Discover (collaborative application access, monitoring, and steering). Furthermore, it enables users to seamlessly access and integrates local and remote services to synthesize customized middleware configurations on demand. Clients can use the Grid as well as Collaboratory services integrated by the middleware to acquire resources, configure and launch applications, connect to monitor and steer the applications, terminate applications and collaborate with other users. A sample application scenario, oil reservoir optimization on the Grid, enabled by the middleware substrate was presented. An experimental evaluation of access latencies for local and remote (over LAN and WAN) Grid services using the middleware substrate was presented. These results show that overheads for using remote services are acceptable.

7. References

[1] W. Allcock, J. Bester, J. Bresnahan, A. Chervenak, L. Liming and S. Tuecke. "GridFTP Protocol Specification," *GGF GridFTP Working Group Document*, September 2002.

[2] J. Bester, I. Foster, C. Kesselman, J. Tedesco and S. Tuecke. "GASS: A Data Movement and Access Service for Wide Area Computing Systems," In *Proceedings Sixth Workshop on I/O in Parallel and Distributed Systems* pages 365-375 Atlanta GA, May 5, 1999.

[3] G. Brose. "JacORB: Implementation and Design of a Java ORB," In *Proceedings. of DAIS'97, IFIP WG 6.1 International Working Conference on Distributed Applications and Interoperable Systems Chapman & Hall,* pages 143-154 September 30-October 2, Cottbus, Germany, 1997.

[4] K. Czajkowski, I. Foster, N. Karonis, C. Kesselman, S. Martin, W. Smith and S. Tuecke. "A Resource Management Architecture for Metacomputing Systems," In *Proceedings IPPS/SPDP '98 Workshop on Job Scheduling Strategies for Parallel Processing,* pages 62-82, Orlando FL, 1998.

[5] K. Czajkowski, S. Fitzgerald, I. Foster and C. Kesselman. "Grid Information Services for Distributed Resource Sharing," In *Proceedings of the Tenth IEEE International Symposium on High-Performance Distributed Computing (HPDC-10), IEEE Press,* San Francisco CA, August 2001.

[6] D. Diachin, L. Freitag, D. Heath, J. Herzog, W. Michels, and P. Plassmann. "Remote Engineering Tools for the Design of Pollution Control Systems for Commercial Boilers," *International Journal of Supercomputer Applications*, 10(2) pages 208-218, 1996.

[7] Event Service Specification, Version 1.1, http://www.omg.org/docs/formal/01-03-01.pdf.

[8] S. Fitzgerald, I. Foster, C. Kesselman, G. von Laszewski, W. Smith and S. Tuecke. "A Directory Service for Configuring High-Performance Distributed Computations," In *Proceedings of 6th IEEE Symposium on High-Performance Distributed Computing (HPDC-6),* pages. 365-375, Portland OR, 1997.

[9] I. Foster and C. Kesselman. Computational Grids, Chapter 2 of "*The Grid: Blueprint for a New Computing Infrastructure*," Morgan-Kaufman, CA 1999.

[10] I. Foster and C. Kesselman. "Globus: A Metacomputing Infrastructure Toolkit," *International Journal of Supercomputer Applications*, 11(2) pages 115-128, 1997.

[11] I. Foster and C. Kesselman, G. Tsudik and S. Tuecke. "A Security Architecture for Computational Grids," *In Proceedings of 5th ACM Conference on Computer and Communications Security Conference,* pages. 83-92, San Francisco CA, 1998.

[12] I. Foster, C. Kesselman, J. Nick and S. Tuecke. "The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration," In *Proceedings of Open Grid Service Infrastructure WG, Global Grid Forum*, 2002.

[13] Global Grid Forum, http://www.gridforum.org/.

[14] R. T. Kouzes, J. D. Myers and W. A. Wulf. "Collaboratories: Doing Science on the Internet," In *Proceedings of IEEE Computer August 1996 IEEE Fifth Workshops on Enabling Technology: Infrastructure for Collaborative Enterprises (WET ICE '96)*, pages 40-46 Stanford CA, June 19-21 1996,

[15] G. von Laszewski, I Foster, J Gawor and P Lane. "A Java Commodity Grid Kit," *Concurrency and Computation: Practice and Experience*, pages 643-662, Volume 13, Issue 8-9, 2001.

[16] V. Mann and M. Parashar. "Engineering an Interoperable Computational Collaboratory on the Grid," *Concurrency and Computation: Practice and Experience, Special Issue on Grid Computing Environments, John Wiley and Sons*, Vol. 14, Issue 13-15, pages. 1569-1593, 2002.

[17] R. Muralidhar and M. Parashar. "An Interactive Object Substrate for Computational Steering of Distributed Simulations," *In Proceedings of the Ninth IEEE International Symposium on High-Performance Distributed Computing (HPDC-9), IEEE Computer Society Press*, pages. 304-305, Pittsburgh PA, August 2000."

[18] Naming Service Specification, Version 1.2, http://www.omg.org/docs/formal/02-09-02.pdf.

[19] G. Olson, D. E. Atkins, R. Clauer, T. Finholt, F. Jahanian, T.L. Killeen, A. Prakash, and T. Weymouth. "The Upper Atmospheric Research Collaboratory," *ACM Interactions*, Vol. 3, pages. 48-55, May-June 1998.

[20] ORB Interoperability Architecture, http://www.omg.org/docs/formal/02-06-17.pdf.

[21] M. Parashar, G. von Laszewski, S. Verma, J. Gawor, K. Keahey, and N. Rehn. "A CORBA Commodity Grid Kit," *Concurrency and Computation: Practice and Experience, Special Issue on Grid Computing Environments, John Wiley and Sons,* Vol. 14, Issue 13-15, pages. 1057-1074, 2002.

[22] Python Globus (pyGlobus), http://www-itg.lbl.gov/gtg/projects/pyGlobus/.

[23] M. Roussos, A. Johnson, J Leigh, C. Barnes, C. Vasilakis, and T. Moher. "The NICE project: Narrative, Immersive, Constructionist/Collaborative Environments for Learning in Virtual Reality," *In Proceedings of ED-MEDIA/ED-TELECOM 97*, pages 917-922, Calgary, Canada, June 1997

[24] M. Russell, G. Allen, G. Daues, I. Foster, T. Goodale, E. Seidel, J. Novotny, J. Shalf, W. Suen, and G von Laszewski. "The Astrophysics Simulation Simulation Collaboratory A Science Portal Enabling Community Software Development," In *Proceedings of Tenth IEEE International Symposium on High Performance Distributed Computing (HPDC-10)*, pages 207-215, San Francisco CA, August 2001.

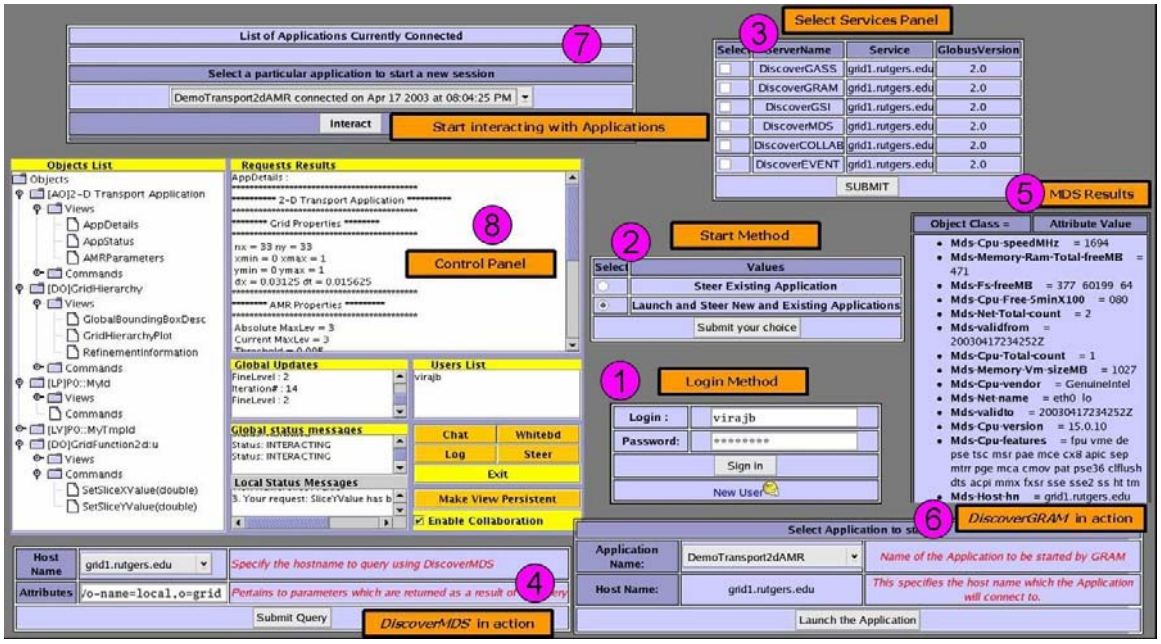
[25] Simple Object Access Protocol (SOAP), 1.1 http://www.w3.org/TR/SOAP/.

[26] M. Thomas, S. Mock and J. Boisseau. "Development of the Web toolkits for Computational Science Portals: The NPACI HotPage," *In Proceedings of 9th IEEE*

International Symposium on the High Performance Distributed Computing (HPDC-9), Pages 308-309, Pittsburgh PA, Aug 14, 2000.

[27] Trading Object Service Specification, Version 1.0, http://www.omg.org/docs/formal/00-06-27.pdf.

Appendix A



DemoTransport2dAMR connected on Apr 17 2003 at 08:04:25 PM

Select	ServerName	Service	GlobusVersion
☐	DiscoverGASS	grid1.rutgers.edu	2.0
☐	DiscoverGRAM	grid1.rutgers.edu	2.0
☐	DiscoverGSI	grid1.rutgers.edu	2.0
☐	DiscoverMDS	grid1.rutgers.edu	2.0
☐	DiscoverCOLLAB	grid1.rutgers.edu	2.0
☐	DiscoverEVENT	grid1.rutgers.edu	2.0

Select	Values
○	Steer Existing Application
●	Launch and Steer New and Existing Applications

Login : virajb
Password: ********

Object Class =	Attribute Value

- Mds-Cpu-speedMHz = 1694
- Mds-Memory-Ram-Total-freeMB = 471
- Mds-Fs-freeMB = 377 60199 64
- Mds-Cpu-Free-5minX100 = 080
- Mds-Net-Total-count = 2
- Mds-validfrom = 20030417234252Z
- Mds-Cpu-Total-count = 1
- Mds-Memory-Vm-sizeMB = 1027
- Mds-Cpu-vendor = GenuineIntel
- Mds-Net-name = eth0 lo
- Mds-validto = 20030417234252Z
- Mds-Cpu-version = 15.0.10
- Mds-Cpu-features = fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm
- Mds-Host-hn = grid1.rutgers.edu

```
Objects
  [AO]2-D Transport Application
    Views
      AppDetails
      AppStatus
      AMRParameters
    Commands
  [DO]GridHierarchy
    Views
      GlobalBoundingBoxDesc
      GridHierarchyPlot
      RefinementInformation
    Commands
  [LP]P0::MyId
    Views
    Commands
  [LV]P0::MyTmpId
  [DO]GridFunction2d:u
    Views
    Commands
      SetSliceXValue(double)
      SetSliceYValue(double)
```

```
AppDetails :
******************************************
********** 2-D Transport Application **********
******************************************
******** Grid Properties ********
******************************************
nx = 33 ny = 33
xmin = 0 xmax = 1
ymin = 0 ymax = 1
dx = 0.03125 dt = 0.015625
******************************************
******** AMR Properties *********
******************************************
Absolute MaxLev = 3
Current MaxLev = 3
```

```
FineLevel : 2
Iteration# : 14
FineLevel : 2
```

Users List: virajb

```
Status: INTERACTING
Status: INTERACTING
```

```
3. Your request: SliceYValue has b
```

Host Name	grid1.rutgers.edu	Specify the hostname to query using DiscoverMDS
Attributes	/o-name=local,o=grid	Pertains to parameters which are returned as a result of query

Application Name:	DemoTransport2dAMR	Name of the Application to be started by GRAM
Host Name:	grid1.rutgers.edu	This specifies the host name which the Application will connect to.

Steps	Sequence of Events
1	Client logs in using “Grid Credentials”
2	Client selects the desired action
3	Client selects and configures local/remote Grid and Collaboratory services
4	Client queries MDS services for resources
5	Client is presented with details of a selected resource
6	Client launches applications on selected resource
7	Client presented with an option to interact with an executing application
8	Client presented with a collaborative interaction and steering interface

Figure 9: A snapshot of the Discover portal.