A MIDDLEWARE ARCHITECTURE FOR INTEGRATING SERVICES ON THE GRID

BY VIRAJ N. BHAT

A thesis submitted to the

Graduate School—New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Master of Science

Graduate Program in Electrical and Computer Engineering

Written under the direction of

Professor Manish Parashar

and approved by

_____

_____

_____

New Brunswick, New Jersey

May, 2003

ABSTRACT OF THE THESIS

A Middleware Architecture for Integrating Services on the Grid

by VIRAJ N. BHAT

Thesis Director: Professor Manish Parashar

The growth of the Internet and the advent of the computational **"Grid"** have made it possible to develop and deploy advanced services to support the infrastructure and applications on the Grid. Recent years have also 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. These services typically have customized architectures and implementation, and build on different enabling technologies. In this thesis 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.

Acknowledgements

I would like to like to acknowledge my **Family in India** who have been supportive of my efforts. I would like to thank my research advisor **Dr. Manish Parashar** for his invaluable guidance, support and encouragement during the course of this work and throughout my graduate studies at Rutgers. I particularly draw my inspiration from his hardwork and enthusiasm with which he goes about this work. I am thankful to **Dr. Deborah Silver** and **Dr. Ivan Marsic** for being on my thesis committee and for their advice and suggestions regarding the thesis. I also acknowledge the suggestions of the committee in developing my technical understanding, research aptitude, thesis writing and presentation skills. I thank my colleagues at **TASSL Lab**, particularly **Manish Agarwal** and **Vincent Matossian** for valuable research discussions, cooperation and collaboration related to this work. I thank the CAIP staff consisting of **Bill Kish**, **Steve Vaccaro** and **James Chun** who would readily help me during times of emergency.

Dedication

To my Parents

Table of Contents

List of Tables

List of Figures

Chapter 1
Introduction

"Grid Computing" [18] 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 [16]. 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 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) [35], Discover [30, 44] and Astrophysics Simulation Collaboratory (ASC) [41]). 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 [21] and the Open Grid Services Architecture (OGSA) [19] is to address these challenges by defining community standards and protocols.

1.1 Contributions

The thesis makes these key contributions:

- To investigate the design and evaluation of a prototype middleware that will enable interoperability between PSE/collaboratory and Grid services and support the overall execution of computational applications on the Grid.

- In this thesis we present the design, implementation and evaluation of the Grid-enabled Discover middleware substrate that enables Grid infrastructure services

provided by Globus Tookit [17] (security, information, resource management, storage) to interoperate with collaboratory services provided by the DISCOVER computational collaboratory (collaborative application access, monitoring, and steering).

- We demonstrate how users can seamlessly access and integrate local and remote services to synthesize customized middleware configurations on demand. This thesis builds on our previous work on the CORBA Community Grid (CoG) Kit [46] and the Discover middleware [31].

- We evaluate the performance of this middleware during the computational applications lifecyle and argue that the concept of using borrowing service does not affect the performance of the middleware significantly.

1.2 Organization

The thesis is organized as follows. Chapter 1 presents background on Computational Collaboratories and Problem Solving environments.It also talks about the services and capabilites they provide. Chapter 2 talks about middleware technologies which are critical for achieving this interoperability as presented in this thesis. Chapter 3 presents our architecture of the Grid enabled middleware substrate. Chapter 4 presents the design and implementation of the Discover Grid enabled middleware substrate. Chapter 5 discusses the operation of the Discover Grid enabled middleware substrate. Chapter 6 discusses the use of the Discover Grid enabled middleware in the process of online Oil Reservoir Optimization. Chapter 7 presents an experimentation evaluation and analysis of of Discover Grid enabled middleware framework. Chapter 8 presents a summary of this thesis and comments on the future work.

Chapter 2
Prior Work

In recent years, there has been considerable research in problem solving environments, computational collaboratories and middleware technologies. In this chapter we summarize recent efforts and the services/capabilities they provide. Our overall goal is to highlight the need for and benefits of integration and interoperability between the services provided by each of these systems. Section 2.1 summarizes the various services provided by computational collaboratories and motivates the need for the Grid enabled middleware. Section 2.2 lists the various technologies required for building this middleware.

2.1 Computational Collaboratories and Problem Solving Environments

Recent efforts aimed at developing and deploying computational collaboratories and problem solving environments to support applications on the Grid include Discover, Astrophysics Simulation Collaboratory (ASC), NPACI HoTPage [45], Upper Atmospheric Research Collaboratory (UARC), Environmental Molecular Sciences Collaboratory (ESML) [28], Diesel Combustion Collaboratory (DCC) [13], Space Physics Aeronomy Research Collaboratory (SPARC) [42] and Narrative-based, Immersive, Constructionist/Collaborative Environments for children (NICE) [39]. Each of these systems provides a set of services to support application development, execution and/or management to their user communities, as summarized in Table 2.1. For

Collaboratories / **Services Offered**	**UARC**	**NICE**	**DCC**	**ESML**	**ASC Portal**	**Discover**
Discovery of Services	**X**	**X**	**X**	**X**	**X**	√
Resource Access	**X**	**X**	**X**	**X**	√	**X**
Resource Notification	**X**	**X**	**X**	**X**	**X**	**X**
Data Movement	*	*	*	*	**X**	**X**
Authentication Authorization	√	√	√	√	√	√
Application Monitoring	√	*	√	√	√	√
Steering	√	√	√	√	√	√
WhiteBoard	√	√	√	√	√	√
Concurency Control	√	√	√	√	√	√
√= services which are present **X= services which are absent** ***= not clear may not be supported**						

Table 2.1: Services Provided by Computational Collaboratories

example, Discover provides services for remote application access, collaborative application monitoring, and controlled application steering. It however lacks services for resource management and allocation and data movement. NICE provides a shared virtual design space for tele-immersive applications but lacks services for resource/information discovery. Clearly, the services required during the lifetime of a Grid application will span multiple such systems, making the interoperability and integration of these systems and their services a desirable feature. The Discover computational collaboratory, which is one of the building blocks for this work, is a virtual meta-laboratory that enables geographically distributed scientists and engineers to collaboratively access, monitor, interact with and control distributed applications using computational portals. The Discover middleware substrate is a peer-to-peer network of Discover interaction and collaboration servers and enables pervasive and collaborative access to geographically distributed applications. Although Discover provides a rich set of services, it clearly needs to interoperate with systems providing Grid services for resource management, application configuration and deployment, data movement, etc. This thesis extends the Discover middleware substrate to be Grid-aware and to enable this interoperability.

2.2 Middleware Technologies

Standardized middleware technologies are critical for achieving the interoperability addressed in this thesis.Several research efforts have emerged to address this issue. Open Grid Services Architecture (OGSA) is a distributed interaction and computing infrastructure providing uniform exposed service semantics (called Grid Services) for creating, naming and discovering transient service instances. Commodity Grid Kits (CoG Kits) [48] are a combination of commodity and Grid technologies to enhance the functionality, maintenance, and deployment of Grid services. These include Java CoG [47], CORBACoG [46] and PythonCoG [38]. CORBACoG (which is also a building block for this work) provides CORBA applications access to the Globus Grid services and support rapid development on the Grid. The use of standard protocols such as IIOP [40] allows interoperability between CORBA [36] based applications and services across operating systems and programming languages. The commercial community is similarly working to enable interoperability between Web Services [9] through the use of standards such as Web Services Definition Language (WSDL) [10], Web Services Flow Language (WSFL) [29], Simple Object Access Protocol (SOAP) [6], Universal Description, Discovery and Integration (UDDI) [3] registry, and XML [7] messaging, to create a platform-independent, open framework for describing, discovering, and integrating services using the Internet. These technologies are rapidly evolving and there is a conscious effort towards integrating business and Grid services and developing a Business Grid [26].

Chapter 3

The Grid Enabled Middleware Architecture

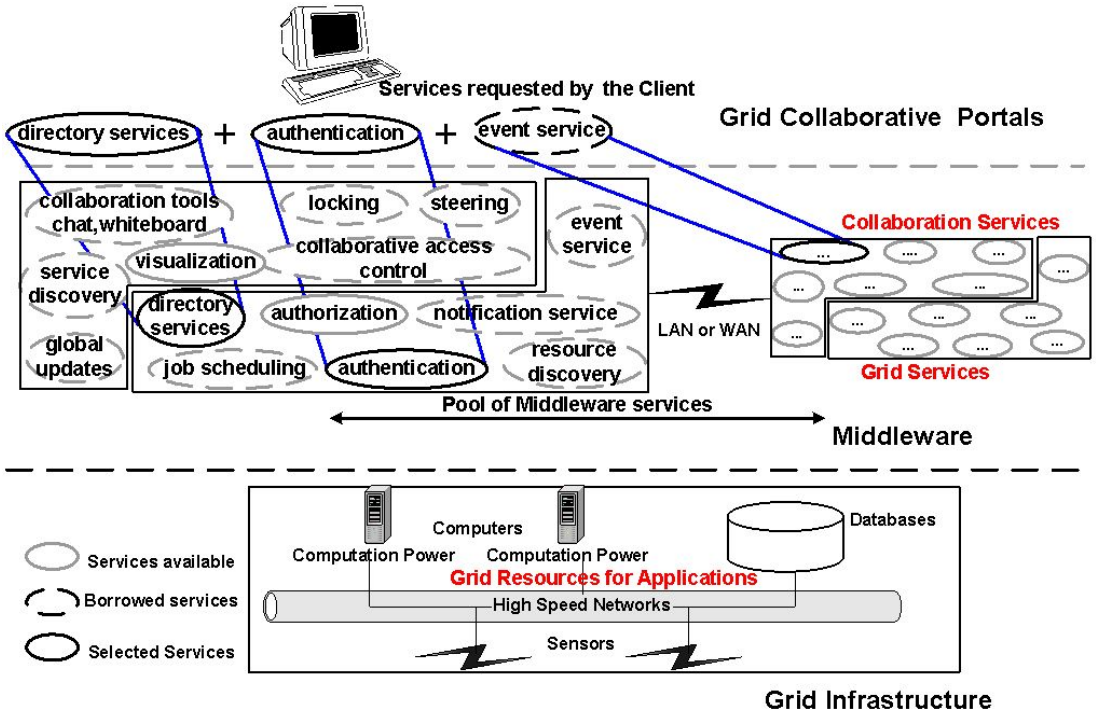

Figure 3.1: Grid Enabled Middleware for Interoperable Collaboratories

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 3.1, 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 3.1). The collaboration

service layer includes services for remote application access, collaborative application monitoring and steering, locking, and concurrency control. 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. 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 that is if certain services are not present at the local host they can be borrowed from a remote host. For example in Figure 3.1, the client is connected to host A(local host) and uses the event and notification services provided by host B (remote host) along with authentication service available locally at host A. The middleware host A borrows services from B and presents a virtual view of these services to the client. The two service layers are explained in Sections 3.1 and Sections 3.2.

3.1 Grid Services

The Grid services layer is composed of key Grid infrastructure services including authorization, authentication, job scheduling, remote job submission, and directory services, which builds on the Globus Toolkit. In addition to these core services, the Grid service layer also provides an event and notification service. The authentication and authorization Grid service is based on the Grid Security Infrastructure (GSI) [20]. It enables services (local and remote) to mutually authenticate with each other. It also enables a service to create and delegate a proxy object on a remote host. The resource naming and directory service is based on Metacomputing Directory Service (MDS) [11, 15] and queries the resource on behalf of a client/service. The job scheduling and submission service is based on Grid Resource Allocation Manager (GRAM) [12]. Job Status can be monitored using the event and notification services. Finally data access/storage service is based on the Global Access to Secondary

Storage (GASS) [4]. The Grid services layer builds on our previous work on the CORBACoG kit [46]. 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. The CORBA Security service authenticates clients/remote hosts and enables them to securely interact with server objects. The CORBA event service is used to implement the event service provided by the Grid enabled Discover middleware.

3.2 Collaboratory Services

The Collaboratory services layer enables clients to collaboratively access Grid applications and to interactively monitor and steer them. It provides services for application discovery and access, access control, collaboration, locking and concurrency control, and logging. It also provides collaboration tools such as whiteboard and chat. Access control services ensure that clients can only access application to which they access privileges and can only interact with them in an authorized way. Locking and concurrency control services ensure that the collaboration proceeds in a controlled way and that the application state is always consistent. The logging service logs all user-user and user-applications interactions. It enables users to replay theirs interactions and enables latecomers to catch up on a collaboration session. Finally, global notification services enable users to obtain real-time updates about the status of an application. The collaboratory services build on the Discover computational collaboratory. The Discover middleware consists of a peer-to-peer network of Discover interaction and collaboration servers and defines collaboratory services across these servers. Discover servers build on commodity web technologies and protocols.

Chapter 4

Design of the DISCOVER Grid Enabled Middleware Substrate

An implementation overview of the Grid-enabled Discover middleware is presented in Figure 4.1. 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 kept as simple as possible to ensure pervasive access. A client connects to its closest host and has access to all (local and remote) 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** [23] and **CORBA Trader** [34] 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 enables scalability and high availability and provides the services necessary to implement the middleware substrate. It also allows interoperability between servers, while allowing them to maintain their individual architectures and implementations. Moreover, servers are typically connected via link with reasonable bandwidth ($\approx$ 1 Mbps). As no assumptions can be made about client-server connections, having the client connect to the "nearest server", and use CORBA/IIOP to connect the server and the desired application may actually reduce

client latencies in some cases. This is because clients (implemented as Java applets) communicate with their "home server" using HTTP and their home server communicates with remote servers on the clients behalf using IIOP. Since IIOP (unlike HTTP), reuses connections and hence reduces connection overheads, it's use over the larger network path helps in reducing client latencies when a large geographical distance separates the two communicating servers, and small chunks of data are transferred ($\approx$ 20Kbytes). This is experimentally demonstrated in [31]. Note that XML based protocols (e.g. SOAP) 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. CORBA also provides sophisticated services such as security, discovery and naming. In the sections to follow we will discuss each component of the Discover in detail.This includes the "**Discover Middleware Host**" Section 4.1 which is referred to as the "**Server**",the services they provide Section 4.2 and the "**DiscoverPortals**" Section 4.3 also known as "**Clients**".

4.1 Discover Middleware Host (Server)

Discover interaction/collaboration servers build on commodity web servers, and extend their functionality (using Java Servlets [25]) to provide specialized services for real-time application interaction and steering and for collaboration between client groups. Clients are Java applets and communicate with the server over HTTP[14] using a series 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 [37] 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

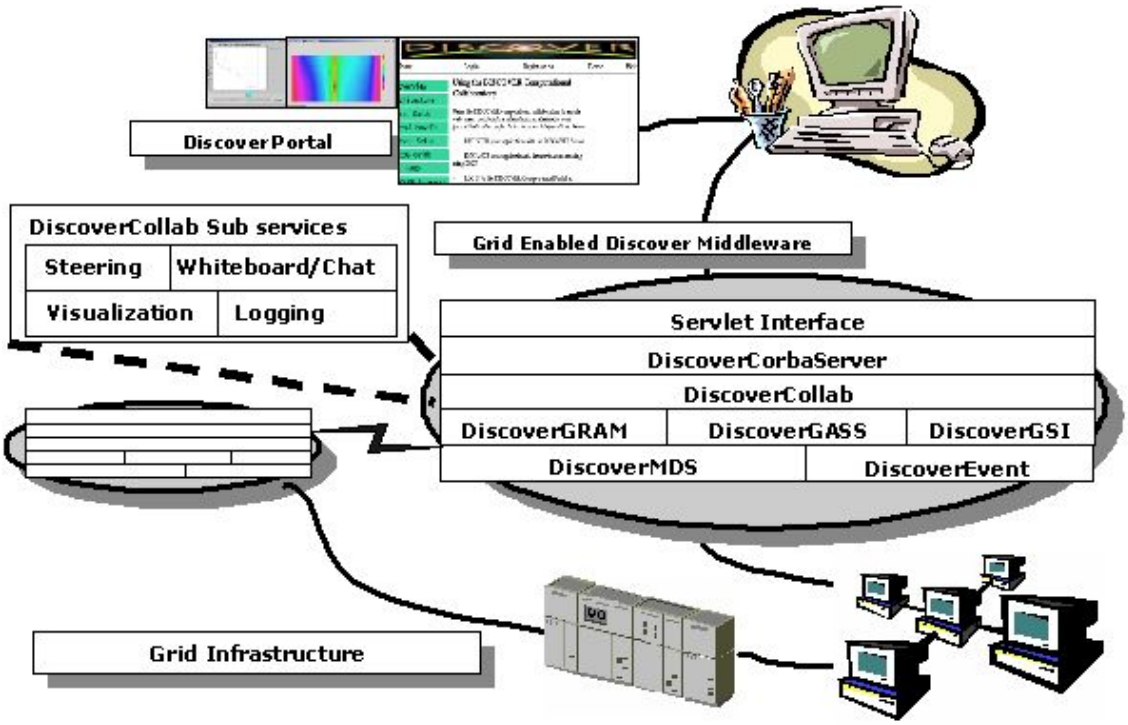

Figure 4.1: Design of the Grid Enabled Middleware

are established between a server and an application:(1) *MainChannel* for application registration and periodic updates, (2)*CommandChannel* for forwarding client interaction requests to the application, (3)*ResponseChannel* for communicating application responses to interaction requests. At the other end, clients differentiate between the various messages (i.e. Response, Error or Update) using Java's reflection mechanism. Core service handlers provided by each server include the *MasterHandler*, *CollaborationHandler*, *CommandHandler*, Security/Authentication , Grid Service Handlers (GSI, MDS, GRAM, GASS) and the Daemon servlet that listens for application connections. Details about the design and implementation of the Discover Interaction and Collaboration servers can be found in [31, 32].

4.2 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 of more of the other interfaces. We will explain each of the services provided by the middleware in detail

A.) ***DiscoverCorbaServer***: 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.

B.) ***DiscoverEvent***: The *DiscoverEvent* interface is also implemented by each host. The *DiscoverEvent* service extends the CORBA Event Service [33] 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 as well as remote channels.

C.) ***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, and uses these credentials to delegate proxy objects.

D.) ***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) [27] libraries. *DiscoverMDS* uses the *DiscoverEvent* service to publish updates to users and other services.

E.) ***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.

F.) ***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 prestage 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 [2] protocol, which also uses the *DiscoverGSI* service.

G.) ***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.

4.3 Discover Portals

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

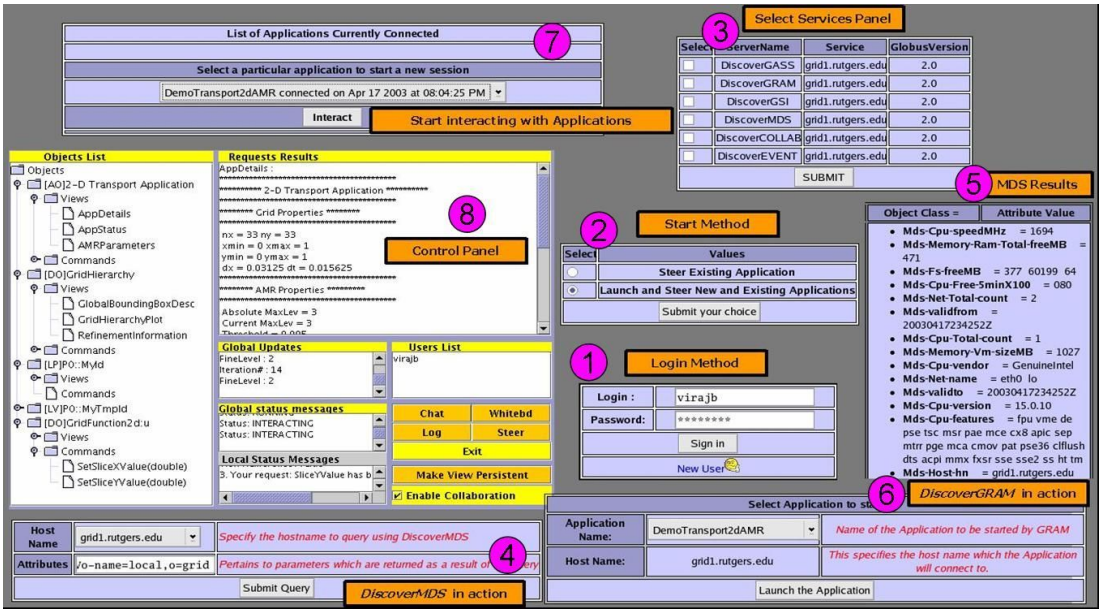

Figure 4.2: Snapshot of the Discover Portal

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. The clients select the set of local or remote services, including resource discovery, application execution, application interrogation, interaction, collaboration, or application/session archival access. After that they would be able to launch the application on their desired choice of resource. The control panel which is downloaded during the process of application interaction provides the user pervasive access to a pool services available on the Grid. For application access, the desktop consists of:

A.) a list of interaction objects and their exported interaction interfaces (views and/or commands). Refer Figure 4.2,

B.) an information pane that displays global updates (current time step of a simulation) from the application,

C.) 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. All users choosing to steer a particular application form a collaboration group by default with a corresponding shared area on the virtual desktop. We have explained the Sequence of Events the User/Client would have to follow in order to "Launch", "Collaborate" and "Terminate" an Application on the "Grid" using our Discover Portal in Table 4.1.

Steps	Sequence of Events
1	Client Logs in with his "Grid Credentials"
2	Chooses the Appropriate Method of Interaction
3	Selects his "Choice" of Grid and Collaboratory Services
4	Queries for the Resource of his Choice
5	Presented with an Elaborate List of Resources
6	Launches the Application on the Resource Selected in "Step 5"
7	Presented with an "Interact" Button to Start Interacting with the Application
8	"Control Panel" is "Downloaded" and Application Interaction with the User Begins

Table 4.1: Sequence of Events Followed by the User during Interaction with the Discover Portal

Chapter 5

Operation of the DISCOVER Grid Enabled Middleware

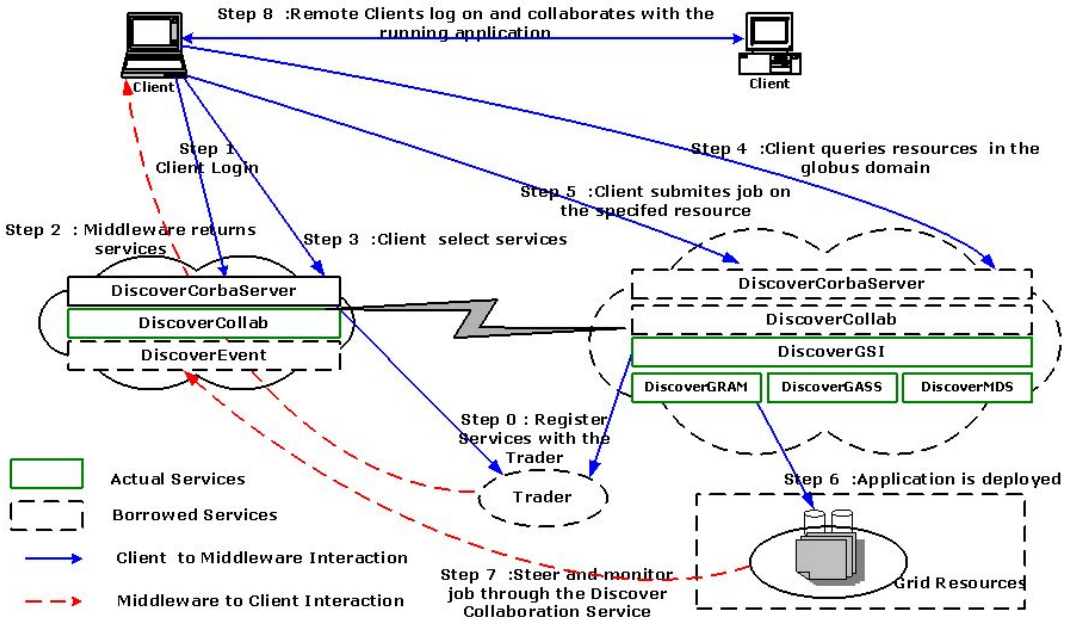

Figure 5.1: Operation of the Discover Grid Enabled Middleware

The overall operation of the Grid enabled middleware is illustrated in Figure 5.1. 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 middleware operations are briefly described below.

5.1 Security and 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:

A.) Client and server objects mutually authenticate using the CORBA Security Service [5].

B.) The client delegates the *DiscoverGSI* server object to create a proxy object that is authorized to communicate with other Grid Services.

C.) 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.

5.2 Registration and Discovery of Services

Services are registered to the CORBA Trader Service at startup. A sample scenario from our example in the Figure. 5.1 the middleware on the left is devoid of the Grid framework and will not have the Grid services registered to the trader entity. All Discover services are identified by their service name and the machine on which they exist thus forming a unique id for each service. The client is presented with an interactive screen to choose the services required for resource discovery, collaboration, job monitoring. Suppose the user decides to use these services remotely the server enables this usage transparently by using commodity distributed technologies like CORBA . The user can then use services available at another middleware to steer and monitor a job.

5.3 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.

5.4 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.

5.5 Distributed Collaboration

The Discover collaboratory enables multiple clients to collaboratively interact with and steer local and remote applications. The *CollaborationHandler* 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 can connect to an application through a remote host, collaboration groups can span multiple hosts. In this case, the *DiscoverCollab* objects at the middleware host poll each other 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 session load now spans multiple hosts.

5.6 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).

5.7 Services Interoperation

As noted in the previous sections the Grid-enabled Discover middleware enables local and remote services to be combined in an ad hoc way and collectively used to achieve desired behaviors. For example, consider the scenario as illustrated in Figure 5.2. In this example, a client copies log files generated by the application during a run using

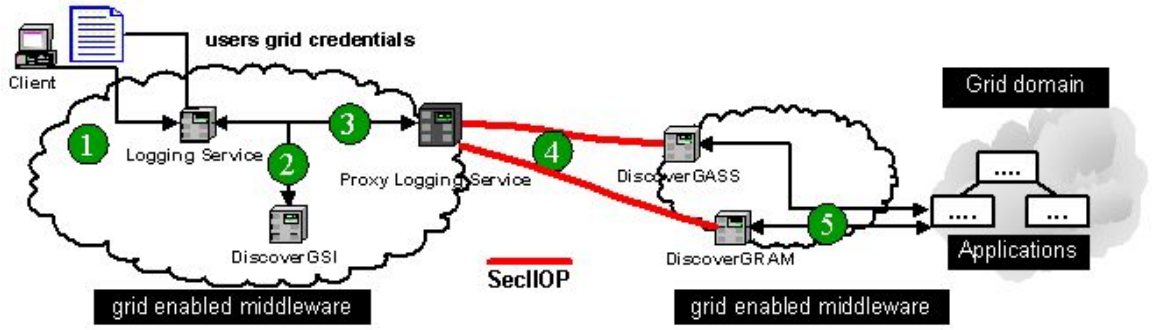

Figure 5.2: Delegation Model Across Services in the Grid Enabled Discover Middleware

a remote *DiscoverGASS* service.

A.) The client logs on to the middleware using the Grid credentials

B.) It accesses the logging collaboratory sub-service (part of *DiscoverCollab*).

C.) The logging service uses the client's credentials and the *DiscoverGSI* service to create and delegate a proxy logging service.

D.) This proxy logging services interacts with the *DiscoverGASS* service to transfer the log files to the local host.

Note that these interactions are over a secure IIOP channel.

Chapter 6

Application Scenario : Oil Reservoir Optimization Using the Grid-Enabled Discover Middleware

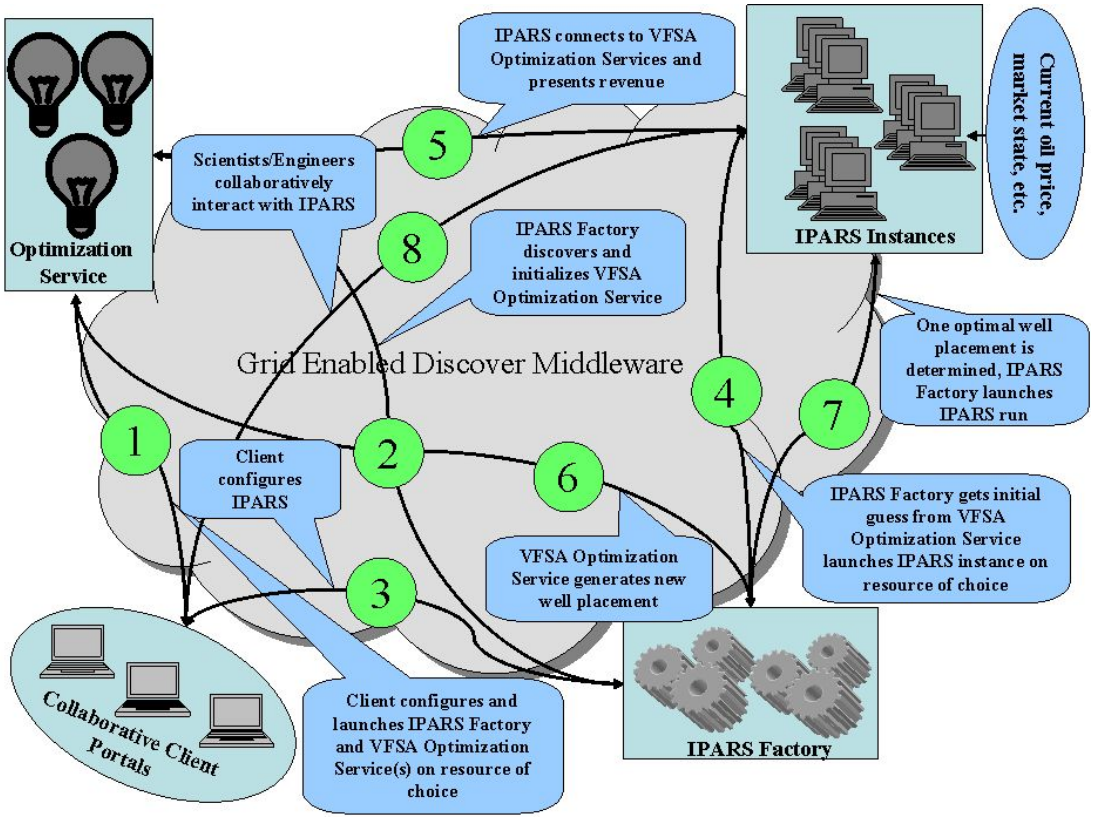

Figure 6.1: Sample Application Scenario: Oil Reservoir Optimization

In this chapter we present a practical utility of our Grid enabled Discover middleware. This sample scenario enables autonomic oil reservoir optimization process on the Grid. The goal of this process is to dynamically optimize the placement and configuration of oil wells to maximize revenue. The overall operation of our middleware in this scenario is illustrated in 6.1. The various entities in this optimization process

include:

- **Integrated Parallel Accurate Reservoir Simulator (IPARS)** [49] providing sophisticated simulation entities that encapsulate complex mathematical models of the physical interaction in the subsurface. These applications execute on distributed heterogeneous systems such as the Grid.

- **IPARS Factory** responsible for configuring and managing multiple instances of IPARS simulations.

- **Very Fast Simulated Annealing (VFSA)** [43] optimization service which is based on statistical physics and the analogy between the model parameters of an optimization problem and particles in an idealized physical system.

- **Grid Enabled Discover middleware** providing Grid and Collaboratory services and enabling resource discovery, resource allocation, job scheduling, job interaction and user collaboration on the Grid.

- **Economic Modeling Service** that uses IPARS simulation outputs and current market parameters (oil prices, costs, etc.) to compute estimated revenues for a particular reservoir configuration.

- **Grid enabled Discover Middleware** providing services for collaboration, resource discovery, resource allocation, job scheduling, and job interaction on the Grid.

- **Discover Collaborative Portals** providing experts with collaborative access to these components. Using these portals, experts can discover and allocate resources, configure and launch peers, and monitor, interact with, and steer peer executions. The portals provide a shared workspace and encapsulate collaboration tools such as Chat and Whiteboard.

6.1 End to End Scenario

The entities involved in the optimization process need to dynamically discover and interact with one another as peers to achieve the overall application objectives. Using the Grid enabled Discover middleware; experts can select desired Grid and Collaboratory services. The Grid services help them (experts) to discover resources on the Grid and launch IPARS, VFSA and Economy Modeling entities on these resources. Then using the DiscoverCollab services they (experts) can monitor, interact and steer these entities. We explain in detail the steps required for the optimization process.

A.) Experts/users logon to the Discover middleware using the Grid credentials. Once they have logged onto the Portal they can choose the list of services they require. If certain services are not present at the current middleware they are borrowed from the nearest middleware in the domain. This enables the users to have a pervasive access to a pool of Grid and Collaboratory services. They (users) use the *DiscoverMDS* service to first locate the resource on a particular host. After querying for resources on a particular host they deploy the IPARS Factory using the *DiscoverGRAM* service which is delegated using the *DiscoverGSI* service. The IPARS Factory discovers and interacts with the VFSA entity to configure and initialize it

B.) The users interact with the IPARS Factory and VFSA using the *DiscoverCollab* service to define application configuration parameters

C.) IPARS Factory utilizes the *DiscoverMDS* service to discover resources on which it can configure and execute IPARS instances. The factory then uses the *DiscoverGRAM* service on the local or remote middleware to execute various instances of the IPARS application. The user can monitor the status of spawned IPARS applications by subscribing to the event channel created by the *DiscoverEvent* service.

D.) The Economic model entity is likewise deployed on the choice of resource selected by the user. This entity interacts with the IPARS simulations to determine current economic values.

E.) The VFSA service which was previously deployed provides the IPARS Factory with optimized well information.

F.) New IPARS simulations are iteratively deployed using the *DiscoverGRAM* service with the optimized well information obtained from the VFSA entity.

G.) These instances of IPARS simulations are collaboratively monitored using the *DiscoverCollab* service. The user has updates of the application using the *DiscoverEvent* service. The user can also transfer log files using the *DiscoverGASS* service. Various sub-services of the *DiscoverCollab* service like chatting, logging and locking are used to steer and manipulate application parameters. Once the optimal well parameters are determined the IPARS Factory configures and deploys a production IPARS run.

H.) The IPARS simulations are terminated using the *DiscoverGRAM* service and the log files are transferred to the local machines for further analysis by experts.

The Discover Grid services is the key to enable the experts to authenticate themselves, establish Grid credentials and appropriately delegate proxies, discover services and resources on the Grid launch the IPARS Factory, IPARS simulations instances, VFSA Optimization service and the Economic Model on these resources, and transfer data and execution logs. The Discover Collaboratory services enable the experts to interactively configure the different entities, to consistently monitor, interact with and steer these entities, and to collaborate with other experts.

Chapter 7
Evaluation

The Grid-enabled Discover middleware is presently deployed at TASSL (The Applied Software Systems Laboratory), 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 [22] Servlet engine and the JacORB [8] an open source implementation of the CORBA ORB.The user signs on to the portal with his userid and password and is presented with a web interface to select services of his interest. The user then searches for the host in the particular domain and the resources available at that point of time using the *DiscoverMDS* service. The portal provides the user with the option to launch an application on a particular host queried in the previous step using the *DiscoverGRAM* service. The user specifies the path of the executable, command line parameters of the application and the host to submit the application. The user receives interactive job updates on the portal using the CORBA event service. The user can also copy files by specifying the source and destination URL using the *DiscoverGASS* service and finally he can terminate the application.

7.1 Performance Evaluation

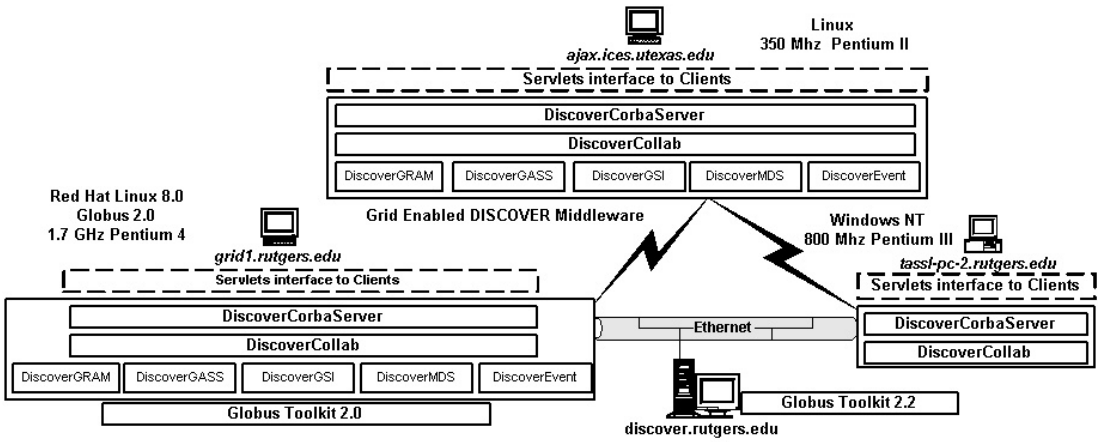

Figure 7.1: Experimental Setup of the Grid Enabled Discover middleware

The overall setup for these experiments is show in Figure 7.1. 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.Local domain in our experiments was the Rutgers University domain and remote and the wide area network consists of services accessed at University of Texas at Austin or visa-versa.

7.2 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 the LAN (**case B-LAN**).

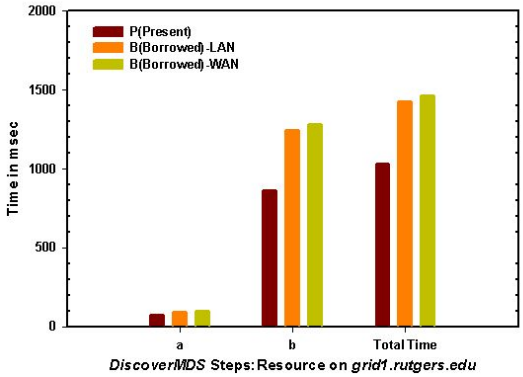

Figure 7.2: *grid1.rutgers.edu* is Queried

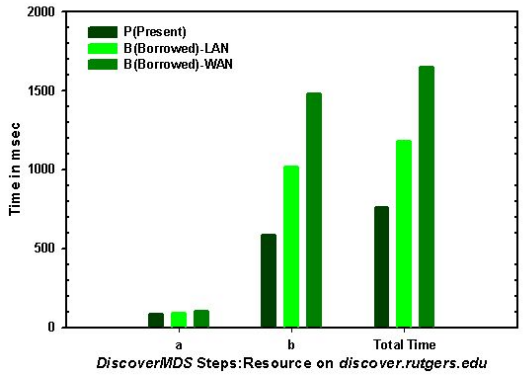

Figure 7.3: *discover.rutgers.edu* is Queried

In the third case the *DiscoverMDS* service is borrowed from a remote host over the WAN (**case B-WAN**). In all three cases clients used the *DiscoverMDS* service to discover resources in **Rutgers** domain. In each case, the experiment consists of two steps:

(a) discovering the *DiscoverMDS* service using the CORBA Trader service

(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 7.2 and 7.3 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.

7.3 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*.

a	**Resolving Services:***DiscoverCollab*
b	**Delegation:***DiscoverGSI*
c	**Event Channel Creation:***DiscoverEvent*
d	**Job Start time on***grid1.rutgers.edu*
e	**Total time to start job: a+b+c+d**
f	**Resolving Services:***DiscoverGRAM*
g	**Delegation:***DiscoverGSI*
h	**Event Channel Creation:***DiscoverEvent*
i	**Job Cancellation Time:***DiscoverGRAM*
j	**Total time to cancel job: f+g+h+i**

Table 7.1: Steps Involved in Starting and Terminating Jobs using the *DiscoverGRAM* Service

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

(d) launch the application on the selected host e.g. *grid1.rutgers.edu*. Application termination similarly consisted of the following steps:

(e) discovering the *DiscoverGRAM* service,

(f) using *DiscoverGSI* to delegate a service proxy,

(g) create an event channel for application monitoring,

(h) 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 7.4 and each step is explained in Table 7.1. 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 a 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. We

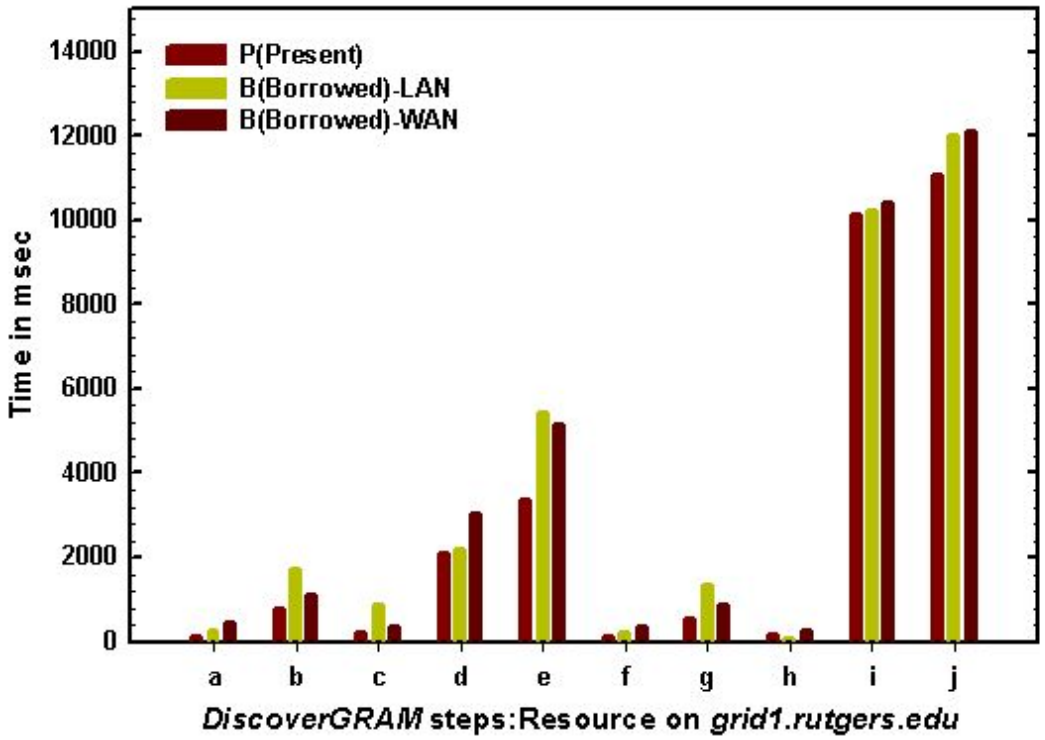

Figure 7.4: *DiscoverGRAM* Launches the *tportamr* Application Using Steps a, b, c and d. It Terminates the Same Application Using Steps f, g, h and i on *grid1.rutgers.edu*.

evaluated the *DiscoverGRAM* service by measuring latencies involved in starting and terminating the ***tportamr*** application on *discover.rutgers.edu* refer Figure 7.5. *discover.rutgers.edu* was connected to *grid1.rutgers.edu* through a local area network(LAN).*discover.rutgers.edu* had Globus Toolkit 2.0 installed. We observe that the latencies involved in starting a job on *discover.rutgers.edu* is quite comparable in the 3 cases mentioned. We were not able to measure the latency for terminating the application in all the three cases due to configuration problems.

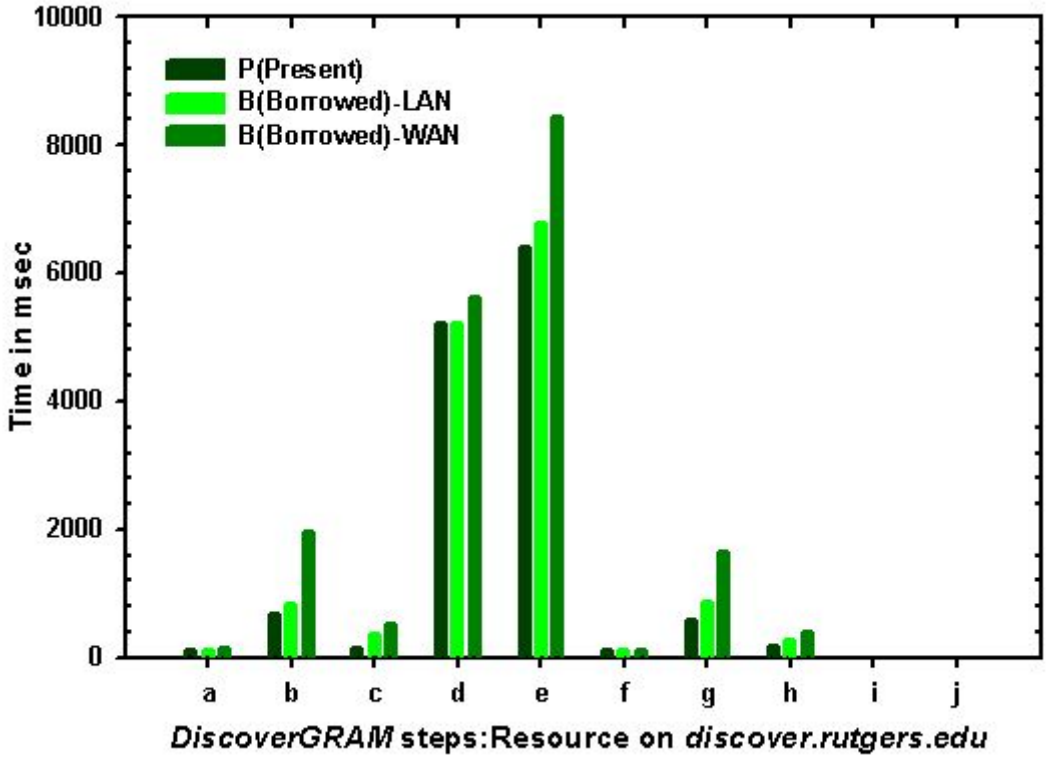

Figure 7.5: *DiscoverGRAM* Launches the *tportamr* Application Using Steps a, b, c and d. It Terminates the Same Application Using Steps f, g, h and i on *discover.rutgers.edu*.

7.4 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 7.6 using a log-log scale. The file sizes and the transfer times varied exponentially and ranged from 2 bytes to ≈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

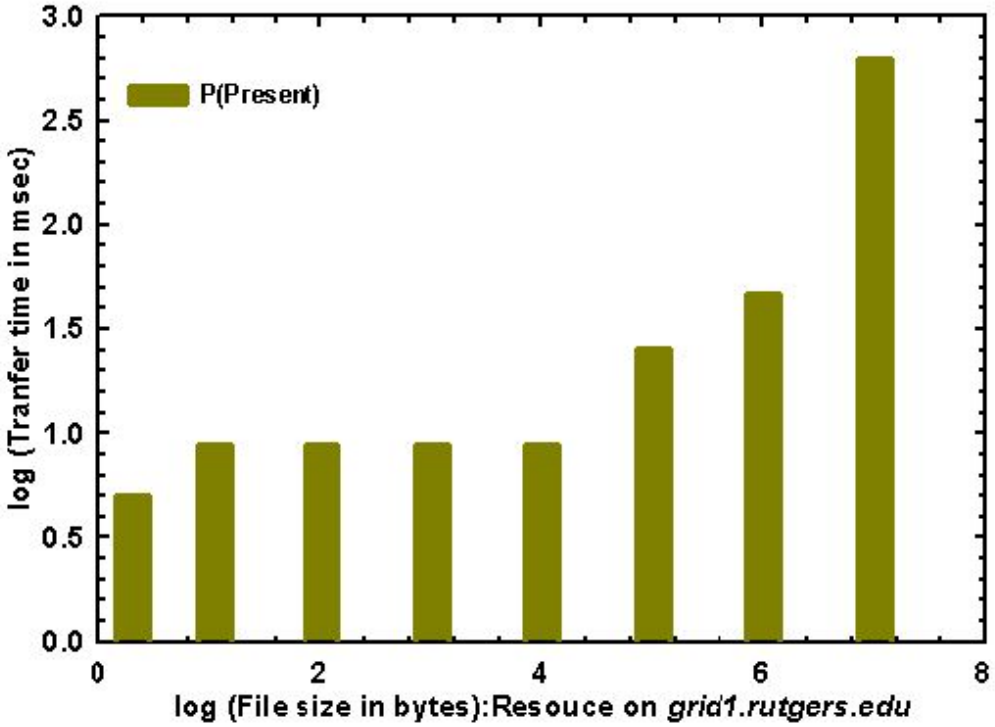

Figure 7.6: Log-Log Plot of Transfer Times for Various File Sizes Using the *DiscoverGASS* Service (P case)

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

7.5 Evalutation 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 [31]. 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 (**case P**), borrowed over a LAN (**case B-LAN**) and borrowed over a WAN (**case B-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.

Chapter 8
Conclusions

This work 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.We are in the process of testing the *DiscoverGASS* services in the **B(Borrowed)-LAN case** and **B(Borrowed)-WAN case**

8.1 Future Work

The Grid enabled Discover middleware is presently built using the services provided by the Globus Toolkit-2. In a related research work [1] we are in the process of enhancing this middleware for supporting "Autonomic" [24] Grid applications.This middleware

will be developed on the **OGSA** (Open Grid Services Architecture) framework which is currently implemented by the Globus Toolkit-3. This next generation middleware will be aligned with the Service Oriented Architecture. The middleware will intelligently manage and execute autonomic applications with huge computational requirements over Grid resources.This middleware will implement key enhancements to the existing Grid middleware and provide services to support Autonomic Grid Applications.This layer uses application context("**Context Awareness**"), high level policies associated with it,information behavior of the application and resource requirements specified by the user. The main components of the this middleware will consist of

- **Autonomic Grid Infrastructure** This infrastructure will build on our Grid enabled middleware which provides Grid and Collaboratory services and enhance these services and align them with the OGSA standards.To enable this behavior our first step will be to build "context awareness", into the middleware which will allow the systems to sense and react to environment/systems.

- **Autonomic Runtime Management** which sets up and configures the application execution environment.This manages and controls all the autonomic requirements (self-optimizing, self healing, self configuring, self protecting).
  This layer uses the context information provided by the infrastructure layer to analyze, execute, plan and monitor the application components.

- **Autonomic Applications**. consists of a new generation of realistic, scientific and engineering simulations of complex physical phenomenon.These Autonomic applications will symbiotically and opportunistically combine computations, experiments and real-time data and will provide important insights to complex physical phenomena. Few of these include "Thermonuclear combustion" ,"Simulation of active flow control of turbulent flows", "Forest fire simulation model" etc.

We would like to allow interoperability between other collaboratories like the "**ASC Portal**","**UARC**" and evaluate their performances in real world scenarios. We believe that such an interoperabilty would allow the reuse and wide usage of already existing sevices, since colaboratories would not be able to implement all the available services to the user community. This would be a great benefit to the user community who could customize their middleware stack on demand.

References

[1] M. Agarwal, V. Bhat, Z. Li, H. Liu, V. Matossian, V. Putty, C. Schmidt, G. Zhang, M. Parashar, B. Khargharia, and S. Hariri. AutoMate: Enabling Autonomic Applications on the Grid. In *Proceedings of Autonomic Computing Workshop The Fifth Annual International Workshop on Active Middleware Services(AMS 2003)*, Seattle, WA (accepted), June 25 2003.

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

[3] T. Bellwood. UDDI (Universal Description Discovery and Integration) Version 2.04 API Specification. http://uddi.org/pubs/ProgrammersAPI-V2.04-Published-20020719.htm, July 19, 2002.

[4] 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 of the Sixth Workshop on I/O in Parallel and Distributed Systems*, pages 365–375, Atlanta, GA, May 5 1999.

[5] B. Blakley. *CORBA Security An Introduction to Safe Computing with Objects*. Addison-Wesley Object Technology Series. Addison Wesley Longman Inc, One Jacob Way, Reading, Massachusetts, 1999.

[6] D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. F. Nielsen, S. Thatte, and D. Winer. Simple Object Access Protocol (SOAP) 1.1. http://www.w3.org/TR/SOAP/, May 08, 2000. W3C.

[7] T. Bray, J. Paoli, and C. Sperberg-McQueen. Extensible Markup Language (XML). REC-xml-19980210, World Wide Web Consortium Recommendation, February 1998.

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

[9] M. Champion, C. Ferris, E. Newcomer, and E. Newcomer. Web Services Architecture. http://www.w3.org/TR/ws-arch/, November 14, 2002.

[10] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. Web Services Description Language (WSDL) 1.1. http://www.w3.org/TR/wsdl, March 15, 2001.

[11] 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*, pages 181–194, San Francisco, CA, August 7-9 2001.

[12] K. Czajkowski, I. Foster, N. Karonis, C. Kesselman, S. Martin, W. Smith, and S. Tuecke. A Resource Management Architecture for Metacomputing Systems. In *Proc. IPPS/SPDP '98 Workshop on Job Scheduling Strategies for Parallel Processing*, pages 62–82, Orlando, Florida, March 30 1998.

[13] 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):208–218, 1996.

[14] R. T. Fielding, J. Gettys, J. C. Mogul, H. F. Nielsen, L. Masinter, P. Leach, and T. B. Lee. Hypertext Transfer Protocol HTTP 1.1. RFC 2616, HTTP Working Group, University of California, Irvine, CA 92717-3425, June 1999.

[15] 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 the 6th IEEE Symposium on High-Performance Distributed Computing*, pages 365–375, Portland, OR, 5-8 August 1997.

[16] I. Foster and A. Iamnitchi. On Death, Taxes, and the Convergence of Peer-to-Peer and Grid Computing. In *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS'03)*, Berkeley, CA, February 20-21 2003.

[17] I. Foster and C. Kesselman. Globus:A Metcomputing Infrastructure Toolkit. *International Journal of Supercomputer Applications*, 11(2):115–128, 1997.

[18] I. Foster and C. Kesselman. *The Grid: Blueprint for a Future Computing Infrastructure*. Morgan Kaufmann Publishers, San Francisco, CA, 1998.

[19] 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 the Open Grid Service Infrastructure WG, Global Grid Forum*, June 22 2002.

[20] I. Foster, C. Kesselman, G. Tsudik, and S. Tuecke. A Security Architecture for Computational Grids. In *Proc. 5th ACM Conference on Computer and Communications Security Conference*, pages 83–92, San Francisco, CA, November 2-5 1998.

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

[22] Apache Tomcat Group. Apache Tomcat. http://jakarta.apache.org/tomcat/. The Apache Jakarta Project.

[23] OMG (Object Management Group). Naming Service Specification Version 1.2. http://www.omg.org/cgi-bin/doc?formal/02-09-02.pdf, September 2002.

[24] P. Horn. Autonomic Computing:IBM's perspective on the State of Information Technology. http://www.research.ibm.com/autonomic/, Oct 2001. IBM Corp.

[25] J. Hunter and W. Crawford. *JAVA Servlet Programming.* O'Reilly & Associates,Inc, Sebastopol, CA-95472,USA., 1998.

[26] IBM. The Era of Grid Computing:Enabling New Possibilities For Your Business. http://www-1.ibm.com/grid/pdf/business_exec_grid.pdf, January 2003. IBM Corp.

[27] Sun Microsystems Inc. Java Naming and Directory Interface(JNDI) 1.2. ftp://ftp.javasoft.com/docs/jndi/1.2/jndi.pdf, July 14,1999.

[28] R. T. Kouzes, J. D. Myers, and W. A. Wulf. Doing Science on the Internet. In *IEEE Computer August 1996,IEEE Fifth Workshops on Enabling Technology:InfraStructure for Collaorative Enterprises(WET ICE 96)*, pages 40–46, Stanford CA, June 19-21 1996.

[29] F. Leymann. Web Services Flow Language (WSFL) 1.0. http://www-3.ibm.com/software/solutions/webservices/pdf/WSFL.pdf, May,2001. IBM Academy of Technology, IBM Software Group.

[30] V Mann, V. Matossian, R. Muralidhar, and M. Parashar. DISCOVER: An Environment for Web-based Interaction and Steering of High-Performance Scientific Applications. *Concurrency and Computation: Practice and Experience, John Wiley and Sons*, 13(8-9):737–754, 2001.

[31] V. Mann and M. Parashar. Middleware Support for Global Access to Integrated Computational Collaboratories. In *Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing,IEEE Computer Society Press*, pages 35–46, San Francisco, CA, August 2001.

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

[33] OMG (Object Management Group). Event Service Specification Version 1.1. http://www.omg.org/cgi-bin/doc?formal/01-03-01.pdf, March 2001.

[34] OMG (Object Management Group). Trading Object Service Specification Version 1.0. http://www.omg.org/cgi-bin/doc?formal/00-06-27.pdf, May 2000.

[35] G. Olson, D. E. Atkins, T. Finholt, and R. Clauer. The Upper Atmospheric Research Collaboratory. *ACM Interactions*, 5(3):48–55, May-June 1998.

[36] A. Pope. *The CORBA Reference Guide Understanding the Common Object Request Broker Architecture*. Addison-Wesley Corporate & Professional. Addison Wesley Longman Inc, One Jacob Way, Reading, Massachusetts, 1999.

[37] J. Postel. Transmission Control Protocol. Std 7,RFC 793, IETF Secretariat c/o Corporation for National Research Initiatives, 1895 Preston White Drive, Suite 100 Reston, VA 20191-5434, September 1981.

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

[39] M. Roussos, A. Johnson, C. Barnes J. Leigh, 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.

[40] W. Ruh, T. Herron, and P. Klinker. *IIOP Complete Understanding CORBA and Middleware Interoperability*. Addison-Wesley Object Technology Series. Addison Wesley Longman Inc, One Jacob Way, Reading, Massachusetts, 1999.

[41] M. Russell, G. Allen, G. Daues, and G. von Laszewski. The Astrophysics Simulation Collaboratory A Science Portal for Enabling Community Software Development. In *Proceedings of the Tenth IEEE International Symposium on High Performance Distributed Computing*, pages 207–215, San Francisco, CA, August 2001.

[42] University of Michigan School of Information. SPARC Space Physics and Aeronomy Research Collaboratory. http://intel.si.umich.edu/sparc, June 2002.

[43] M. K. Sen and P. L. Stoffa. *Global Optimization Methods in Geophysical Inversion*. Advances in Exploration Geophysics 4. Elsevier Science, New York, NY, 1995.

[44] Rutgers University TASSL (The Applied Software Systems Laboratory). Discover Portal. http://www.discoverportal.org.

[45] M. Thomas, S. Mock, and J. Boisseau. Development of the Web Toolkits for Computational Science Portals: The NPACI HotPage. In *Proceedings of the 9th IEEE International Symposium on High Performance Distributed Computing(HPDC 2000)*, pages 308–309, Pittsburgh,PA, Aug 14 2000.

[46] S. Verma, M. Parashar, J. Gawor, and G. von Laszewski. Design and Implementation of a CORBA Community Grid Kit. In *Proceedings of the 2nd*

International Workshop on Grid Computing,Lecture Notes in Computer Science,Editors:C. A. Lee, Springer-Verlag, pages 2–13, Denver, CO, November 2001.

[47] G. von Laszewski, I. Foster, J. Gawor, and P. Lane. A Java Community Grid Kit. *Concurrency and Computation: Practice and Experience*, 13(8-9):643–662, 2001.

[48] G. von Laszewski, I. Foster, J. Gawor, W. Smith, and S. Tuecke. CoG Kits: A Bridge between Commodity Distributed Computing and High-Performance Grids. In *Proceedings of the ACM Java Grande 2000 Conference*, pages 97–106, San Francisco, CA, June 3-5 2000.

[49] J. A. Wheeler and M. Peszyńska. IPARS: Integrated Parallel Reservoir Simulator. http://www.ticam.utexas.edu/CSM. Center for Subsurface Modeling, University of Texas at Austin.