A Middleware Substrate for Integrating Services on the Grid

Viraj. N. Bhat (virajb@caip.rutgers.edu)
Rutgers, The State University of New Jersey, The Applied Software Systems Laboratory, Department of Electrical and Computer Engineering, Rutgers, The State University of New Jersey, 94 Brett Road, Piscataway, NJ 08854.

Manish. Parashar (parashar@caip.rutgers.edu)
Rutgers, The State University of New Jersey, The Applied Software Systems Laboratory, Department of Electrical and Computer Engineering, Rutgers, The State University of New Jersey, 94 Brett Road, Piscataway, NJ 08854.

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.

Keywords: Grid Computing, Middleware, Service Interoperability and Computational Collaboratories

1. Introduction

Grid computing [17, 15] 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 multipurpose domain services for authentication, authorization, discovery, messaging, data input/output, and application/domain specifi c services such as application

Support for this work was provided by the NSF via grant numbers ACI 9984357 (CAREERS), EIA 0103674 (NGS) and EIA-0120934 (ITR), DOE ASCI/ASAP (Caltech) via grant numbers PC295251 and 82-1052856.

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) [29], Discover [40] and Astrophysics Simulation Collaboratory (ASC) [36, 2]. 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.

Enabling collaboratories/PSEs to share services and capabilities has many advantages. However realizing such interoperability also presents many challenges. This is because 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 [20] and the Open Grid Services Architecture (OGSA) [18] is to address these challenges by defining community standards and protocols.

The primary contribution of this paper is the design, implementation and evaluation 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 Grid-enabled Discover middleware substrate that enables Grid infrastructure services provided by the Globus Toolkit [16] 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. The use of the prototype middleware to enable the optimization of an oil reservoir using decentralized Grid services is presented. This work builds on our previous work on the CORBA Community Grid (CoG) Kit [42] and Discover middleware [27].

The rest of this paper is organized as follows. Section 2 presents background and related work. Section 3 presents the overall architecture of the middleware. Section 4 presents the implementation of the middleware. Section 5 describes the operation of the middleware. Section 6 describes the use of the middleware for Oil Reservoir Optimization on the Grid. Section 7 presents an experimentation evaluation of the middleware. Section 8 presents some conclusions of this work. Appendix DiscoverPortal gives a graphical illustration of our Discover Portal with sequence of steps to be followed by the user during interaction with the Discover Portal.

2. Related Work

In recent years, there has been considerable work in problem solving environments, computational collaboratories and middleware technologies. In this section we summarize selected 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 these systems.

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 [41], Upper Atmospheric Research Collaboratory (UARC), Environmental Molecular Sciences Collaboratory (ESML) [23], Diesel Combustion Collaboratory (DCC) [12], Space Physics Aeronomy Research Collaboratory (SPARC) [37] and Narrative-based, Immersive, Constructionist/Collaborative Environments for children (NICE) [34]. 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 I. For 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 confi guration and deployment, data movement, etc. This research extends the Discover middleware substrate to be Grid-aware and to enable this interoperability.

Table I. A Summary of Services Provided by PSEs/Computational Collaboratories

Collaboratories / Services Offered	UARC	NICE	DCC	ESML	ASC Portal	Discover
Discovery of Services	**X**	**X**	**X**	**X**	**X**	**P**
Resource Access	**X**	**X**	**X**	**X**	**P**	**X**
Resource Notifi cation	**X**	**X**	**X**	**X**	**X**	**X**
Data Movement	*	*	*	*	**X**	**X**
Authentication Authorization	**P**	**P**	**P**	**P**	**P**	**P**
Application Monitoring	**P**	*	**P**	**P**	**P**	**P**
Steering	**P**	**P**	**P**	**P**	**P**	**P**
WhiteBoard	**P**	**P**	**P**	**P**	**P**	**P**
Concurency Control	**P**	**P**	**P**	**P**	**P**	**P**

P = services which are present
X = services which are absent
*** = not clear may not be supported**

2.2. *Middleware Technologies*

Middleware technologies are critical for achieving the interoperability addressed in this paper. 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) [44], which complement the OGSA philosophy of enabling services for community use, are a combination of commodity and Grid technologies to enhance the functionality, maintenance, and deployment of Grid services. These include Java CoG [43], CORBACoG [42] and PythonCoG [24]. 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 [35] allows interoperability between CORBA based applications and services across operating systems and programming languages.
The commercial community is similarly working to enable interoperability between Web Services [8] through the use of standards such as Web Services Definition Language (WSDL) [9], Web Services Flow Language (WSFL) [25], 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 [22].

3. The Grid-enabled Middleware Architecture

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 (see Figure 1): the ***Collaboratory Service Layer*** consisting of Discover Collaboratory services and the ***Grid Service Layer*** consisting of Globus Grid services. The *Collaboratory Service Layer* (Refer to subsection 3.2) includes services for remote application access, collaborative application monitoring and steering, locking, and concurrency control. The *Grid Service Layer*(Refer to subsection 3.1) includes infrastructure services such as resource discovery, authentication, security, directory

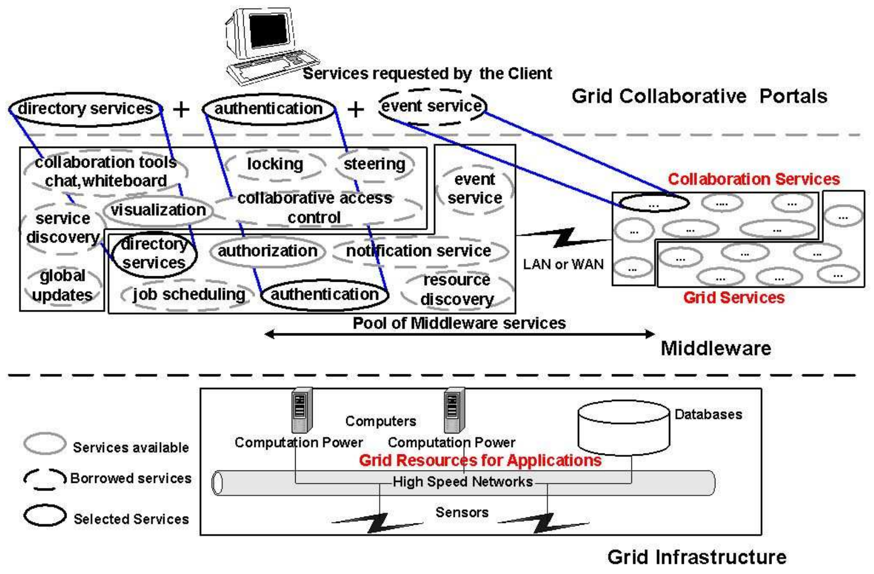


Figure 1. Discover Grid-enabled middleware for interoperable collaboratories

services, resource management and scheduling. Some services, such as the event service, span both layers. Services in both service layers can be accessed by 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.

3.1. *Grid Services Layer*

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 notifi cation service. The authentication and authorization Grid service is based on the Grid Security Infrastructure (GSI) [19]. 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) [14, 10] and queries the resource on behalf of a client/service. The job scheduling and submission service is based on Grid Resource Allocation Manager (GRAM) [11]. Job Status can be monitored using the event and no-

tifi cation 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 [42]. The CORBACoG provides access to CORBA server objects, which are wrappers around Globus Grid services. It also provides access to the CORBA Security [5] and the CORBA Event Service [30]. 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 Layer*

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 notifi cation 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 defi nes collaboratory services across these servers. Discover servers build on commodity web technologies and protocols [26].

4. Implementation of the Discover Grid-enabled Middleware Substrate

An implementation overview of the Discover's Grid-enabled middleware is presented in Figure 2. It consists of collaborative client portals at the front end, computational resources, services and applications(DIOS) [28] 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 confi gurations 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

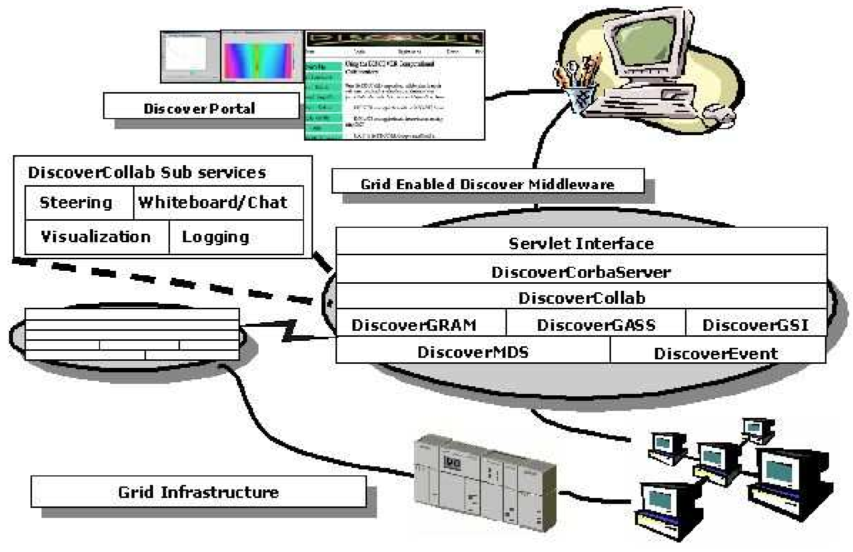


Figure 2. Implementation of the Discover Grid Enabled Middleware

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 [35] and provides peer-to-peer connectivity between hosts within and across domains. Server/service discovery mechanisms are built using the CORBA Naming [32] and Trader [31] 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 (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 [13]), 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 (<20Kbytes). This is experimentally demonstrated in [26]. 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 [33] based object marshalling. CORBA also provides sophisticated services such as security, discovery and naming.

4.1. *Discover Middleware Host(Server)*

Discover interaction/collaboration servers build on commodity web servers, and extend their functionality (using Java Servlets [21]) 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 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: (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*, *Command Handler*, *Security/Authentication Handler*, *Grid Service Handlers* (GSI, MDS, GRAM, GASS) and a daemon servlet that listens for application connections. Details about the design and implementation of the Discover Interaction and Collaboration servers can be found in [27, 26].

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.

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 [30] and enables users/services to monitor the status of applications and resources. The service defi nes an event channel at each host and clients/services can publish and subscribe to local as well as 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, and uses these credentials 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) [39] libraries. *DiscoverMDS* uses the *DiscoverEvent* service to publish updates to users and other services.

DiscoverGRAM: The *DiscoverGRAM* service represents the GRAM service provided by globus 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.

4.3. *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 (refer to Appendix on DiscoverPortal). 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 (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. All users choosing to steer a particular application form a collaboration group by default with a corresponding shared area on the virtual desktop.

5. Operation of the Discover Grid enabled Middleware

The 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 (through the local *DiscoverCorbaSever* service). Each service is uniquely identified by trader by its name and the machine address

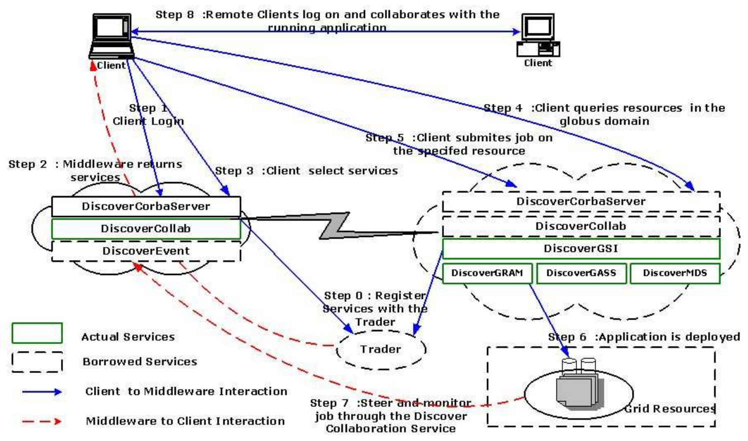

Step 8 :Remote Clients log on and collaborates with the running application
Client
Client
Step 1 Client Login
Step 4 :Client queries resources in the globus domain
Step 5 :Client submites job on the specifed resource
Step 2 : Middleware returns services
Step 3 :Client select services
DiscoverCorbaServer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

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

of its host. A client logging onto 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.

5.1. 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 (CORBA GSI Server Object) between the client and the service. Each Discover server supports a two-level access control for collaboratory services, the fi rst level manages access to the server while the second level manages access to a particular application. Applications are required to be registered and provide a list of users and their access privileges. This information is used for customized access control.

5.2. 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 identifi ed by the service-id “Discover”. The service offer contains the CORBA object reference and a list of properties defi ned as name-value pairs. Thus the object can be identifi ed based on the service it provides or its properties. Applications are located using their globally unique identifi ers, 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.

5.3. 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 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 fi les using the specifi ed protocol.

5.4. 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.

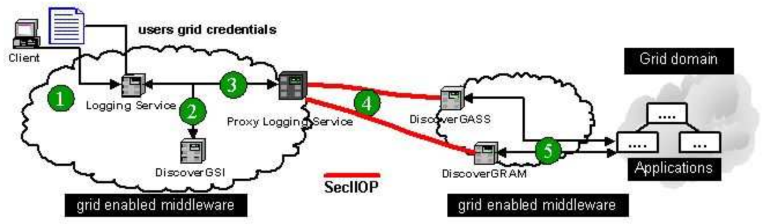


Figure 4. Delegation model across services.

5.5. 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). 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.

6. Application Scenario: Oil Reservoir Optimization using the Grid-enabled Discover Middleware

In this section we briefly describe the use of the Grid-enabled Discover middleware to enable the online optimization of an oil reservoir. The goal of this

Grid application is to dynamically optimize the placement and configuration of oil wells using remote optimization services to maximize revenue. The overall operation of this application is summarized in Figure 5. The entities involved in the optimization process include: (1) **Integrated Parallel Accurate Reservoir Simulator (IPARS)** [45] providing sophisticated simulation components that encapsulate complex mathematical models of the physical interaction in the subsurface, and execute on distributed computing systems on the Grid. (2) **IPARS Factory** responsible for configuring IPARS simulations, executing them on resources on the Grid and managing their execution. (3) **Very Fast Simulated Annealing (VFSA)** [38] optimization service based on statistical physics and the analogy between the model parameters of an optimization problem and particles in an idealized physical system. (4) **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. (5) **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. (6) **Discover Collaborative Portals** providing experts (scientists, engineers) with collaborative access to other peer 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.

The entities involved in the optimization process need to dynamically discover and interact with one another as peers to achieve the overall application objectives. The experts use the portals to interact with the Discover middleware to discover and allocate appropriate resource, and to deploy the IPARS Factory, VFSA and Economic model peers ((1)). The IPARS Factory discovers and interacts with the VFSA service peer to configure and initialize it ((2)). The expert interacts with the IPARS Factory and VFSA to define application configuration parameters ((3)).

The IPARS Factory then interacts with the Discover middleware to discover and allocate resources and to configure and execute IPARS simulations ((4)). The IPARS simulation now interacts with the Economic model to determine current revenues, and discovers and interacts with the VFSA service when it needs optimization ((5)). VFSA provides IPARS Factory with optimized well information ((6)), which then launches new IPARS simulations ((7)). Experts can discover and collaboratively monitor and interactively steer IPARS simulations, configure the other services and drive the scientific discovery process ((8)). Once the optimal well parameters are determined, the IPARS Factory configures and deploys a production IPARS run.

The Discover Grid services enable the experts to authenticate themselves, establish Grid credentials and appropriately delegate proxies, discover services

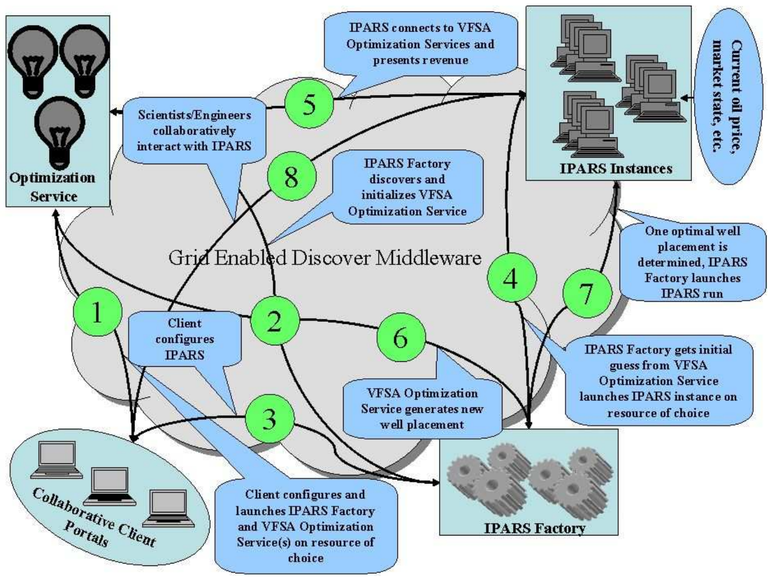

IPARS connects to VFSA Optimization Services and presents revenue
IPARS Instances
Current oil price, market state, etc.
Optimization Service
Scientists/Engineers collaboratively interact with IPARS
IPARS Factory discovers and initializes VFSA Optimization Service
Grid Enabled Discover Middleware
One optimal well placement is determined, IPARS Factory launches IPARS run
Client configures IPARS
IPARS Factory gets initial guess from VFSA Optimization Service launches IPARS instance on resource of choice
VFSA Optimization Service generates new well placement
Collaborative Client Portals
Client configures and launches IPARS Factory and VFSA Optimization Service(s) on resource of choice
IPARS Factory
1
2
3
4
5
6
7
8

Figure 5. Overview of the Oil Reservoir Optimization Application using the Discover Grid-enabled Middleware.

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 confi gure the different entities, to consistently monitor, interact with and steer these entities, and to collaborate with other experts.

7. Experimental Evaluation

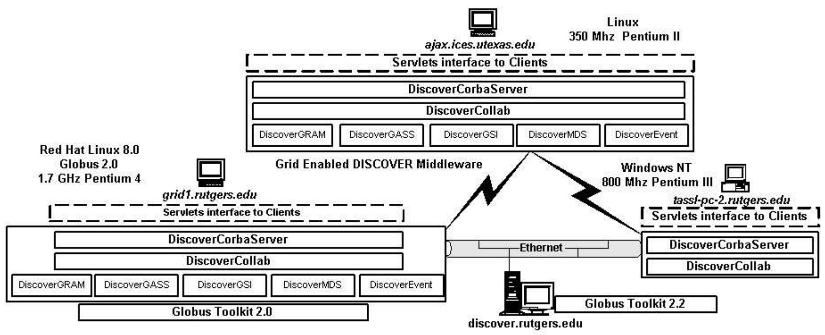
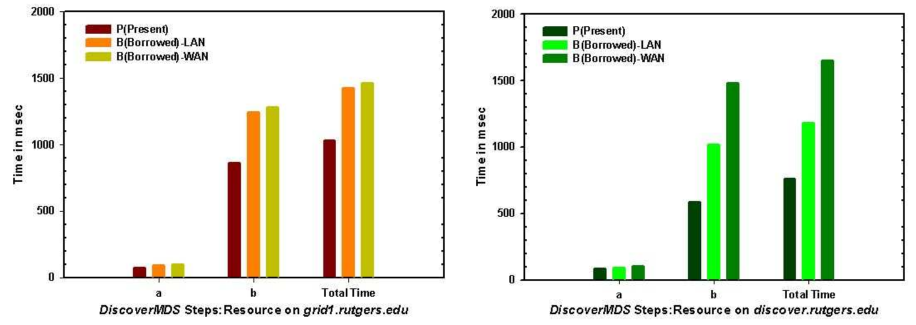


Figure 6. Experimental Setup of the Grid-enabled Discover Middleware

This section presents an experimental evaluation of the Grid-enabled Discover middleware. Experiments consisted of deployments at Rutgers University and at University of Texas. Deployments at *grid1.rutgers.edu* and *ajax.ices.utexas.edu* had installations of Grid and Collaboratory services while *discover.rutgers.edu* had only Grid services and *tassl-pc-2.rutgers.edu* had only Collaboratory services(Refer to Figure 6). 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. Note that an evaluation of the Collaboratory Services was presented in [26].



a. *grid1.rutgers.edu* b. *discover.rutgers.edu*

Figure 7. DiscoverMDS service discovers and queries resources on local machine and machine in the LAN **a** is the time to locate the *DiscoverMDS* service and **b** is the time to query the resources on the selected host.

7.1. 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 7.a. and 7.b. 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.

7.2. 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 8. 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 launching and terminating the application are quite comparable for the three cases. The large termination time is due to the cleanup performed by GRAM.

a	**Resolving services: *DiscoverGRAM***
b	**Delegation : *DiscoverGSI***
c	**Event channel creation: *DiscoverEvent***
d	**Job start time on *grid1.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)**

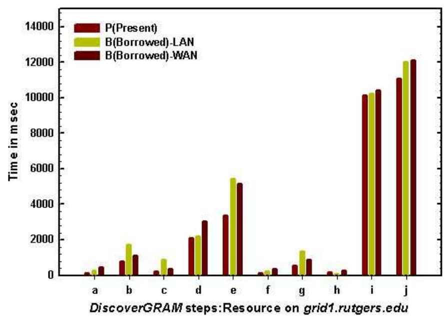


Figure 8. *DiscoverGRAM* service launches the *tportamr* application using steps a, b, c, and d, and terminates the *tportamr* application using steps f, g, h, i on *grid1.rutgers.edu*.

7.3. 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 [27]. 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.

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*. This experiment considers the case P where the *DiscoverGASS* service was locally present.

The file transfer times and the file sizes in bytes are plotted in Figure 9 using a log-log scale. The file sizes varied exponentially from 2 bytes to 10 MB. The equivalent file transfer times ranged from 9 msec. to 637 msec. It is seen that the *DiscoverGASS* performed well for medium file sizes (9 msec. for 2 bytes and 47 msec. for 1 MB). However the performance deteriorated (637 msec.) as file sizes approached 10 MB. The size of log files generated during the course of 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).

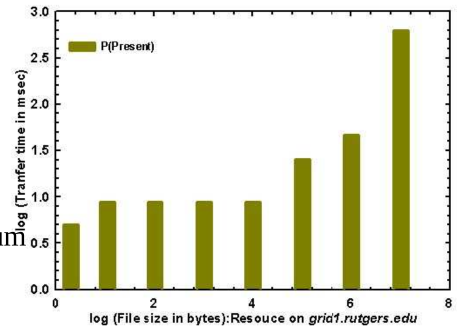


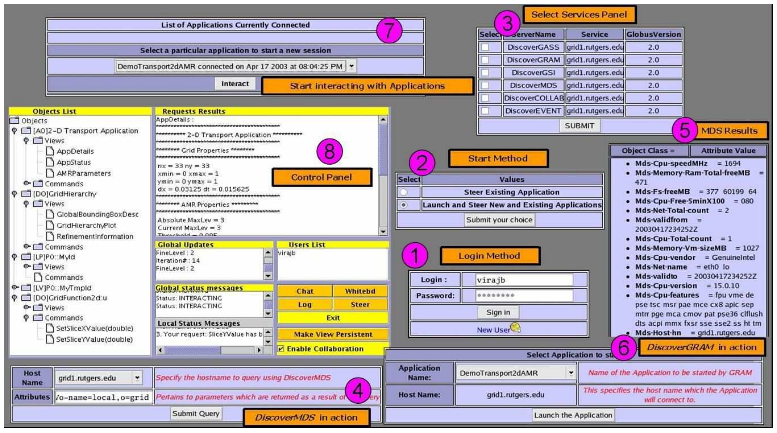
Figure 9. Log-Log plot of transfer times for various file sizes using DiscoverGASS service (P case).

8. 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 integrate local and remote services to synthesize customized middleware confi gurations on demand. Clients can use the Grid as well as Collaboratory services integrated by the middleware to acquire resources, confi gure 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.

Appendix

DiscoverPortal



List of Applications Currently Connected

Select a particular application to start a new session

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

Objects List

```
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)
```

Requests Results

```
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
```

Global Updates

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

Users List: virajb

Global status messages

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

Local Status Messages

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

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

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 ery*

Select Application to st

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.*

Figure 10. Snapshot of the Discover Portal

Table II. Sequence of Events Followed by the User during Interaction with the Discover Portal

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**

References

1. Allcock, W., J. Bester, J. Bresnahan, A. Chervenak, L. Liming, S. Meder, and S. Tuecke.: September 2002, ‘GridFTP Protocol Specification’. GridFTP Working Group Document, GGF.

2. Allen, G., T. Goodale, and E. Seidel: 1999, ‘The Cactus Computational Collaboratory:Enabling Technologies for Relativistic Astrophysics, and a Toolkit for Solving PDEs by Communities in Science and Engineering.’. In: *Proceedings of the 7th Symposium on Frontiers of Massively Parallel Computation(Frontiers99),IEEE*. Annapolis, MD, pp. 36–43.
3. Bellwood, T.: July 19, 2002, ‘UDDI (Universal Description Discovery and Integration) Version 2.04 API Specification.’. http://uddi.org/pubs/ProgrammersAPI-V2.04-Published-20020719.htm.
4. Bester, J., I. Foster, C. Kesselman, J. Tedesco, and S. Tuecke.: 1999, ‘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*. Atlanta, GA, pp. 365–375.
5. Blakley, B.: 1999, *CORBA Security An Introduction to Safe Computing with Objects*, Addison-Wesley Object Technology Series. One Jacob Way, Reading, Massachusetts: Addison Wesley Longman Inc.
6. Box, D., D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. F. Nielsen, S. Thatte, and D. Winer: May 08, 2000, ‘Simple Object Access Protocol (SOAP) 1.1.’. http://www.w3.org/TR/SOAP/. W3C.
7. Bray, T., J. Paoli, and C. Sperberg-McQueen: February 1998, ‘Extensible Markup Language (XML)’. REC-xml-19980210, World Wide Web Consortium Recommendation.
8. Champion, M., C. Ferris, E. Newcomer, and E. Newcomer: November 14, 2002, ‘Web Services Architecture.’. http://www.w3.org/TR/ws-arch/.
9. Christensen, E., F. Curbera, G. Meredith, and S. Weerawarana: March 15, 2001, ‘Web Services Description Language (WSDL) 1.1.’. http://www.w3.org/TR/wsdl.
10. Czajkowski, K., S. Fitzgerald, I. Foster, and C. Kesselman: 2001, ‘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, pp. 181–194.
11. Czajkowski, K., I. Foster, N. Karonis, C. Kesselman, S. Martin, W. Smith, and S. Tuecke.: 1998, ‘A Resource Management Architecture for Metacomputing Systems.’. In: *Proc. IPPS/SPDP ’98 Workshop on Job Scheduling Strategies for Parallel Processing*. Orlando, Florida, pp. 62–82.
12. Diachin, D., L. Freitag, D. Heath, J. Herzog, W. Michels, and P. Plassmann: 1996, ‘Remote Engineering Tools for the Design of Pollution Control Systems for Commercial Boilers.’. *International Journal of Supercomputer Applications* **10**(2), 208–218.
13. Fielding, R. T., J. Gettys, J. C. Mogul, H. F. Nielsen, L. Masinter, P. Leach, and T. B. Lee.: June 1999, ‘Hypertext Transfer Protocol HTTP 1.1’. RFC 2616, HTTP Working Group, University of California, Irvine, CA 92717-3425.
14. Fitzgerald, S., I. .Foster, C. Kesselman, G. von Laszewski, W. Smith, and S. Tuecke: 1997, ‘A Directory Service for Configuring High-Performance Distributed Computations’. In: *Proceedings of the 6th IEEE Symposium on High-Performance Distributed Computing*. Portland, OR, pp. 365–375.
15. Foster, I. and A. Iamnitchi: 2003, ‘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.
16. Foster, I. and C. Kesselman: 1997, ‘Globus:A Metcomputing Infrastructure Toolkit.’. *International Journal of Supercomputer Applications* **11**(2), 115–128.
17. Foster, I. and C. Kesselman: 1998, *The Grid: Blueprint for a Future Computing Infrastructure*. San Francisco, CA: Morgan Kaufmann Publishers.

18. Foster, I., C. Kesselman, J. Nick, and S. Tuecke: 2002, '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*.
19. Foster, I., C. Kesselman, G. Tsudik, and S. Tuecke.: 1998, 'A Security Architecture for Computational Grids.'. In: *Proc. 5th ACM Conference on Computer and Communications Security Conference*. San Francisco, CA, pp. 83–92.
20. Global Grid Forum, 'Global Grid Forum.'. http://www.gridforum.org.
21. Hunter, J. and W. Crawford: 1998, *JAVA Servlet Programming*. Sebastopol, CA-95472,USA.: O'Reilly & Associates,Inc.
22. IBM: January 2003, 'The Era of Grid Computing:Enabling New Possibilities For Your Business.'. http://www-1.ibm.com/grid/pdf/business_exec_grid.pdf. IBM Corp.
23. Kouzes, R. T., J. D. Myers, and W. A. Wulf: 1996, 'Doing Science on the Inter-net.'. In: *IEEE Computer August 1996,IEEE Fifth Workshops on Enabling Technology:InfraStructure for Collaorative Enterprises(WET ICE 96)*. Stanford CA, pp. 40–46.
24. Lawrence Berkeley National Laboratory, 'Python Globus (pyGlobus).'. http://www-itg.lbl.gov/gtg/projects/pyGlobus.
25. Leymann, F.: May, 2001, 'Web Services Flow Language (WSFL) 1.0.'. http://www-3.ibm.com/software/solutions/webservices/pdf/WSFL.pdf. IBM Academy of Technology, IBM Software Group.
26. Mann, V. and M. Parashar: 2001, '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*. San Francisco, CA, pp. 35–46.
27. Mann, V. and M. Parashar: 2002, 'Engineering an Interoperable Computational Collaboratory on the Grid'. *Concurrency and Computation: Practice and Experience, Special Issue on Grid Computing Environments* **14**(13-15), 1569–1593.
28. Muralidhar, R. and M. Parashar: 2000, '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*. Pittsburgh, PA, pp. 304–305.
29. Olson, G., D. E. Atkins, T. Finholt, and R. Clauer: 1998, 'The Upper Atmospheric Research Collaboratory'. *ACM Interactions* **5**(3), 48–55.
30. OMG (Object Management Group): March 2001, 'Event Service Specification Version 1.1.'. http://www.omg.org/cgi-bin/doc?formal/01-03-01.pdf.
31. OMG(Object Management Group): May 2000, 'Trading Object Service Specification Version 1.0.'. http://www.omg.org/cgi-bin/doc?formal/00-06-27.pdf.
32. OMG(Object Management Group): September 2002, 'Naming Service Specification Version 1.2.'. http://www.omg.org/cgi-bin/doc?formal/02-09-02.pdf.
33. Pope, A.: 1999, *The CORBA Reference Guide Understanding the Common Object Request Broker Architecture*, Addison-Wesley Corporate & Professional. One Jacob Way, Reading, Massachusetts: Addison Wesley Longman Inc.
34. Roussos, M., A. Johnson, C. B. J. Leigh, C. Vasilakis, and T. Moher: 1997, 'The NICE Project: Narrative, Immersive, Constructionist/Collaborative Environments for Learning in Virtual Reality'. In: *Proceedings of ED-MEDIA/ED-TELECOM 97*. Calgary, Canada, pp. 917–922.
35. Ruh, W., T. Herron, and P. Klinker: 1999, *IIOP Complete Understanding CORBA and Middleware Interoperability*, Addison-Wesley Object Technology Series. One Jacob Way, Reading, Massachusetts: Addison Wesley Longman Inc.
36. Russell, M., G. Allen, G. Daues, and G. von Laszewski: 2001, 'The Astrophysics Simulation Collaboratory A Science Portal for Enabling Community Software De-

velopment.'. In: *Proceedings of the Tenth IEEE International Symposium on High Performance Distributed Computing*. San Francisco, CA, pp. 207–215.

37. School of Information, University of Michigan: June 2002, 'SPARC Space Physics and Aeronomy Research Collaboratory'. http://intel.si.umich.edu/sparc.
38. Sen, M. K. and P. L. Stoffa: 1995, *Global Optimization Methods in Geophysical Inversion*, Advances in Exploration Geophysics 4. New York, NY: Elsevier Science.
39. Sun Microsystems.: July 14,1999, 'Java Naming and Directory Interface(JNDI) 1.2.'. ftp://ftp.javasoft.com/docs/jndi/1.2/jndi.pdf.
40. TASSL (The Applied Software Systems Laboratory), R. U., 'Discover Portal.'. http://www.discoverportal.org.
41. Thomas, M., S. Mock, and J. Boisseau: 2000, '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)*. Pittsburgh,PA, pp. 308–309.
42. Verma, S., M. Parashar, J. Gawor, and G. von Laszewski: 2001, '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*. Denver, CO, pp. 2–13.
43. von Laszewski, G., I. Foster, J. Gawor, and P. Lane: 2001, 'A Java Community Grid Kit.'. *Concurrency and Computation: Practice and Experience* **13**(8-9), 643–662.
44. von Laszewski, G., I. Foster, J. Gawor, W. Smith, and S. Tuecke: 2000, 'CoG Kits: A Bridge between Commodity Distributed Computing and High-Performance Grids'. In: *Proceedings of the ACM Java Grande 2000 Conference*. San Francisco, CA, pp. 97–106.
45. Wheeler, J. A. and M. Peszyńska, 'IPARS: Integrated Parallel Reservoir Simulator'. http://www.ticam.utexas.edu/CSM. Center for Subsurface Modeling, University of Texas at Austin.