A Middleware Substrate for Integrating Services on the Grid *

Viraj Bhat and Manish Parashar

The Applied Software Systems Laboratory,
Department of Electrical and Computer Engineering,
Rutgers, The State University of New Jersey,
94 Brett Road, Piscataway, NJ 08854
{virajb, parashar}@caip.rutgers.edu
http://www.caip.rutgers.edu/TASSL

Abstract. In this paper we present the design, implementation and evaluation of the Grid-enabled Discover 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 configurations on demand.

1 Introduction

Grid computing [1, 2] 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 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) [3], Discover [4], Astrophysics Simulation Collaboratory (ASC) [5], Diesel Combustion Collaboratory (DCC) [6], and Narrative-based, Immersive, Constructionist/Collaborative Environments for children (NICE) [7]). These systems provide specialized services to their user

* 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 1052856.

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 and the Open Grid Services Architecture (OGSA) [1] is to address these challenges by defining community standards and protocols.

The primary objective of this paper is to investigate the design of a prototype middleware that will enable interoperability between PSE/collaboratory and Grid services to support the overall execution of computational applications on the Grid. In this paper we present the design, implementation and evaluation of the Grid-enabled Discover middleware substrate that enables Grid infrastructure services provided by the Globus Toolkit [2] to interoperate with collaboratory services provided by the Discover computational collaboratory, and enables users to seamlessly access and integrate local and remote services to synthesize customized middleware configurations on demand. This work builds on our previous work on the CORBA Community Grid (CoG) Kit [8] and Discover [4].

2 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 mid-

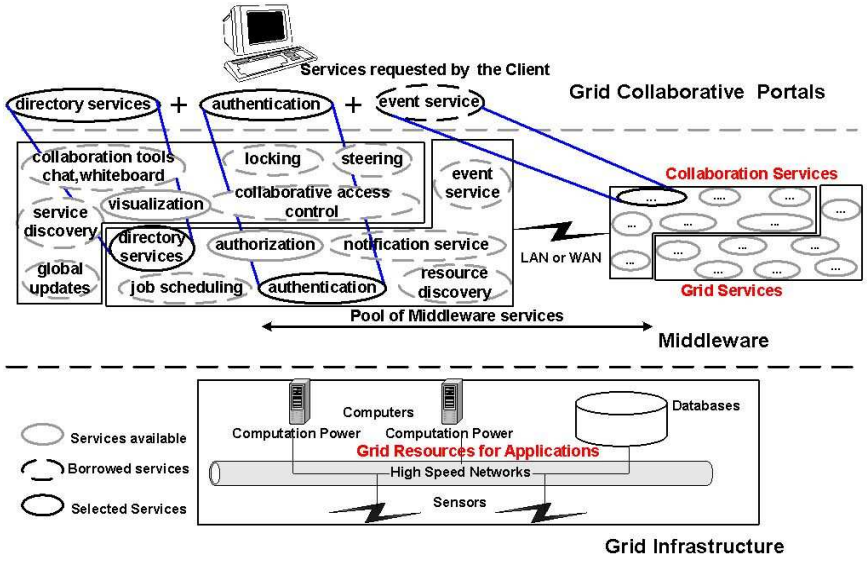


Fig. 1. Discover Grid-enabled middleware for interoperable collaboratories

dleware 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 ***Grid Service Layer*** consisting of Globus Grid services, and the ***Collaboratory Service Layer*** consisting of Discover collaboratory services.

3 The Grid-enabled Middleware Architecture

The Grid-enabled Discover middleware architecture 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. These components are described below. The prototype implementation of the 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 and Trader services, which allows a server to locate remote servers and to access applications/services connected to the remote servers.

3.1 Discover Middleware Host (Server)

Discover interaction/collaboration servers build on commodity web servers, and extend their functionality (using Java Servlets) to provide specialized services

for Grid resource access, application deployment, management, real-time application interaction and steering for collaboration between client groups. Clients are Java applets and communicate with the server over HTTP. Application-to-server communication uses a standard distributed object protocols such as CORBA or a more optimized, custom protocol over TCP sockets. An *ApplicationProxy* object is created for each active application/service at the server, and is given a unique identifier. This object encapsulates the entire context for the application. Three communication channels are established between a server and an application for application registration and updates, client interaction requests and application responses respectively. Core service handlers provided by each server include the *MasterHandler*, *CollaborationHandler*, *CommandHandler*, Security/Authentication Handler, Grid Service Handlers (GSI, MDS, GRAM, GASS) and the Daemon servlet that listens for application connections. Details about the Discover servers can be found in [4].

3.2 Discover Middleware Services

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

3.3 The Discover Portal

The Discover portal consists of a virtual desktop with local and shared areas. The shared areas implement a replicated shared workspace and enable collaboration among dynamically formed user groups. Locking mechanisms are used to maintain consistency. The base portal is presented to the user after authentication and access verification using Grid credentials. This provides the user with a list of available Grid and Collaboratory services that the user is authorized to access and allows the user to select the set of local or remote services to be used during the session. The application interaction desktop consists of (1) a list of interaction objects and their exported interaction interfaces (views and/or commands), (2) an information pane that displays global updates (current time step of a simulation) from the application, and (3) a status bar that displays the current mode of the application (computing, interacting) and the status of

issued command/view requests. The list of interaction objects is once again customized to match the client's access privileges. Chat and whiteboard tools can be launched from the desktop to support collaboration. View requests generate separate (possibly shared) panes using the corresponding view plug-in.

4 Operation of the Discover Grid enabled Middleware

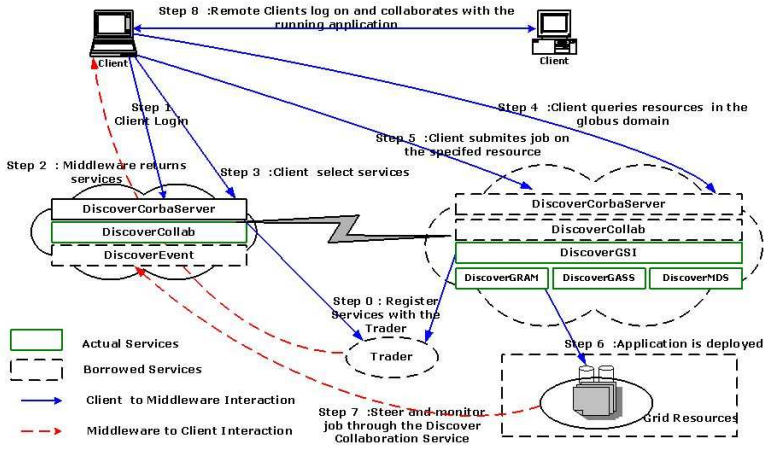


Fig. 2. Operation of the Discover Grid-enabled middleware.

This overall operation of the Grid enabled middleware is illustrated in Figure 2. 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 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.

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 first 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.

Discovery of servers, applications and resources: Peer Discover servers locate each other using the CORBA trader services. The CORBA trader service maintains server references as service-offer pairs. All Discover servers are identified by the service-id "Discover". The service offer contains the CORBA object reference and a list of properties defined as name-value pairs. Thus the object can be identified based on the service it provides or its properties. Applications are located using their globally unique identifiers, which are dynamically assigned by the Discover server and are a combination of the server's IP address and a local count at the server. Resources are discovered using the Globus MDS Grid information service, which is accessed via the *MDSHandler* servlet and the *DiscoverMDS* service interface.

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

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

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 en-

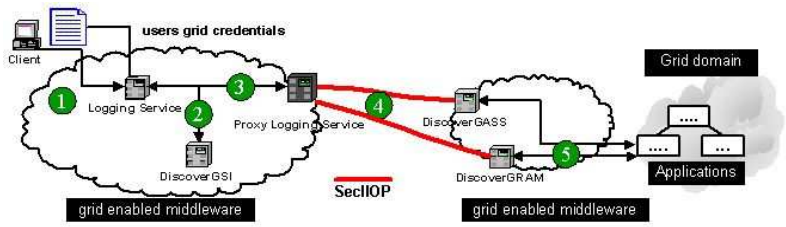


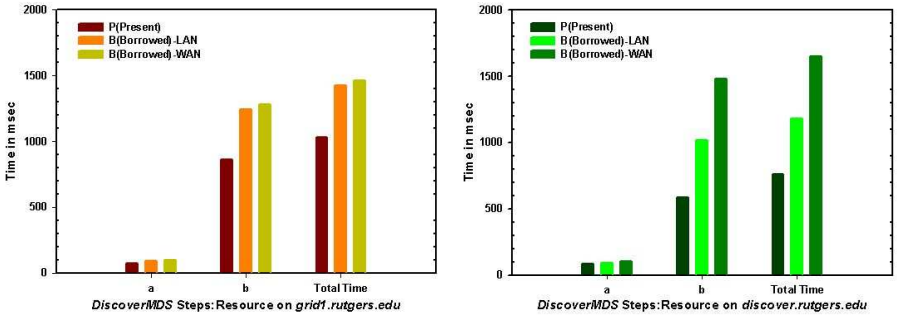
Fig. 3. Delegation model across services.

ables 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 3. In this example, a client copies log files generated by the application during a run using a remote *DiscoverGASS* service. The client logs on to the middleware (step 1) and access the logging collaboratory service (part of DiscoverCollab). The logging service uses the client's credentials and the *DiscoverGSI* service (step 2) to create and delegate a proxy logging service (step 3). This proxy logging services interacts with the *DiscoverGASS* service to transfer the log files to the local host (step 4). Note that these interactions are over a secure IIOP channel.

5 Experimental Evaluation

This section presents an experimental evaluation of the Discover middleware.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. Note that an evaluation of collaboratory services was presented in [4].



(a) *DiscoverMDS* queries resources on *grid1.rutgers.edu*

(b) *DiscoverMDS* queries resources on *discover.rutgers.edu*

Fig. 4. *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.

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 4(a) and 4(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.

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 5. 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)

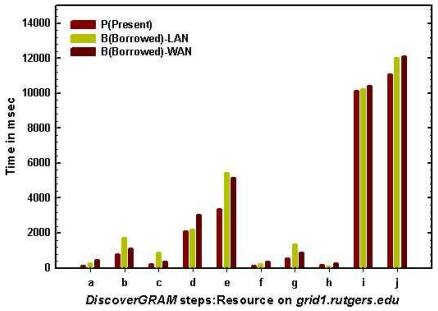


Fig. 5. *DiscoverGRAM* service launches the *tportamr* application using steps a, b, c, and d, and terminates the *tportamr* application using steps f, g, h, i on *grid1.rutgers.edu*.

Evaluation of the 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 6 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 small and 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 eval-

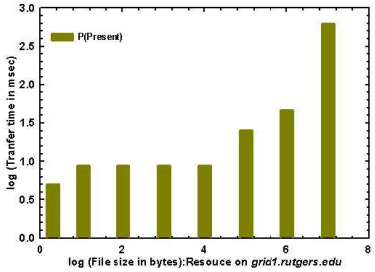


Fig. 6. Log-Log plot of transfer times for various file sizes using *DiscoverGASS* service (P case).

uating cases where the service is borrowed over LAN (B-LAN) and over WAN (B-WAN).

6 Conclusions

This paper presented the design, implementation, operation and evaluation of the Discover Grid-enabled middleware substrate. The middleware substrate enables Grid infrastructure services provided by the Globus Toolkit (security, information, resource management, storage) to interoperate with collaboratory services provided by Discover (collaborative application access, monitoring, and steering). Furthermore, it enables users to seamlessly access and integrate 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.

References

1. Foster, I., Kesselman, C., Nick, J., Tuecke, S.: 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. (2002)
2. Foster, I., Kesselman, C.: Globus:A Metcomputing Infrastructure Toolkit. International Journal of Supercomputer Applications **11** (1997) 115–128
3. Olson, G., Atkins, D.E., Finholt, T., Clauer, R.: The Upper Atmospheric Research Collaboratory. ACM Interactions **5** (1998) 48–55
4. Mann, V., Parashar, M.: 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 (2001) 35–46
5. Russell, M., Allen, G., Daues, G., von Laszewski, G.: 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, San Francisco, CA (2001) 207–215
6. Diachin, D., Freitag, L., Heath, D., Herzog, J., Michels, W., Plassmann, P.: Remote Engineering Tools for the Design of Pollution Control Systems for Commercial Boilers. International Journal of Supercomputer Applications **10** (1996) 208–218
7. Roussos, M., Johnson, A., J. Leigh, C.B., Vasilakis, C., Moher, T.: The NICE Project: Narrative, Immersive, Constructionist/Collaborative Environments for Learning in Virtual Reality. In: Proceedings of ED-MEDIA/ED-TELECOM 97, Calgary, Canada (1997) 917–922
8. Verma, S., Parashar, M., Gawor, J., von Laszewski, G.: 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 (2001) 2–13