CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. 2001; volume (number): 000–000

A CORBA Commodity Grid Kit

Gregor von Laszewski,[2] Manish Parashar,[1]
Snigdha Verma,[1] Jarek Gawor,[2] Kate Keahey[2], and Nell Rehn[2]

[1]*The Applied Software Systems Laboratory, Department of Electrical and Computer Engineering, Rutgers, The State University of New Jersey, 94 Brett Road, Piscataway, NJ 08854-8058, U.S.A.*
[2]*Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439, U.S.A.*

Submitted to:

Journals Production Department, John Wiley & Sons, Ltd.,
Chichester, West Sussex, PO19 1UD, U.K.

SUMMARY

This paper reports on an ongoing research project aimed at designing and deploying a CORBA Commodity Grid (CoG) Kit. The overall goal of this project is to enable the development of advanced Grid applications while adhering to state-of-the-art software engineering practices and reusing the existing Grid infrastructure. As part of this activity, we are investigating how CORBA can be used to support the development of Grid applications. In this paper, we outline the design of a CORBA Commodity Grid Kit that will provide a software development framework for building a CORBA "Grid domain." We also present our experiences in developing a prototype CORBA CoG Kit that supports the development and deployment of CORBA applications on the Grid by providing them access to the Grid services provided by the Globus Toolkit.

KEY WORDS: Grid Computing, CORBA, Globus, DISCOVER, Java CoG Kit

1. INTRODUCTION

The past decade has seen the emergence of computational Grids: infrastructures aimed at allowing the programmer to aggregate powerful and sophisticated resources scattered around the globe. To enable this goal, the Grid computing community has concentrated on the creation of advanced services that allow access to high-end remote resources such as batch systems at supercomputing centers, large-scale storage systems, large-scale instruments, and remote applications. This effort has resulted in the development of Grid services that enable application developers to authenticate, access, discover, manage, and schedule remote Grid resources, but that are often incompatible with

Copyright © 2001 John Wiley & Sons, Ltd.
Received 23 July 2001
Revised 30 Nov. 2001

commodity technologies. As a result, it is difficult to integrate these services into the software engineering process adopted by most application developers.

At the same time, considerable advances have been made in developing and refining commodity technologies for distributed computing. One such effort is the Common Object Request Broker Architecture (CORBA) defined by the Object Management Group (OMG), an independent consortium of vendors. CORBA specifies an open, vendor independent and language independent architecture for distributed application development and integration. Furthermore, CORBA defines a standard interoperability protocol (i.e. GIOP and IIOP) that enables different CORBA implementations and applications to interoperate and be portable across vendors. Key features of CORBA such as high-level distributed computing model, vendor independence, and a strong interoperability thrust makes it an attractive and popular distributed computing standard and a serious candidate to be considered by application developers as part of the Grid infrastructure.

Recently, a number of research groups have started to investigate Commodity Grid Kits (CoG Kits) in order to explore the affinities of the Grid and commodity technologies. Developers of CoG Kits have the common goal of developing mappings and interfaces between Grid services and a particular commodity technology. We believe that CoG Kits will encourage and facilitate the use of Grid technologies, while at the same time leveraging the benefits of the underlying commodity technologies. Currently, CoG Kits are being developed for the Java platform [28, 29], Java Server Pages [6], Python [8], and Perl. This paper describes our experiments in defining the CORBA CoG Kit that allows CORBA applications to access (and provide) services on the Grid. Such an integration would provide a powerful application development environment for high-end users and would create a CORBA "Grid domain."

In this paper we first give a brief overview of the Grid and its architecture and introduce the services and protocols that we intend to integrate in our CORBA CoG Kit [27]. We then briefly outline the requirements, advantages, and disadvantageous of CORBA technologies from the viewpoint of Grid developers. Next, we propose an architecture that supports access to Grid functionality as part of our CORBA CoG Kit, and we present the design, implementation, and application of a prototype. In the final section, we present our conclusions and identify areas for further activities.

2. THE GRID

Copyright © 2001 John Wiley & Sons, Ltd.

The term “Grid” has emerged in the past decade to denote an integrated distributed computing infrastructure for advanced science and engineering applications. The fundamental Grid concept is based on coordinated resource sharing and problem solving in dynamic multi-institutional virtual organizations . Besides access to a diverse set of remote resources, services, and applications among different organizations, Grid computing is required to facilitate highly flexible sharing relationships among these organizations ranging from client-server to peer-to-peer. An example of a typical Grid client-server relationship is a supercomputer centre in which a client submits jobs to the supercomputer batch queue. An example of peer-to-peer computing is the collaborative online steering of high-end (distributed) applications and advanced instruments [4, 23, 30].

Grids must support different levels of control ranging from fine-grained access control to delegation and from single user to multi-user and collaborations, and different services such as scheduling, co-allocation, and accounting. These requirements are not sufficiently addressed by the current commodity technologies, including CORBA. Although sharing of information and communication between resources are allowed, it is not easy to coordinate the use of distributed resources spanning multiple institutions and organizations. The Grid community has developed protocols, services and tools, which address the issues arising from sharing resources in peer communities. The community is also addressing security solutions that support management of credentials and policies when computations span multiple institutions, secure remote access to compute and data resources, and information query protocols that provide services for obtaining the configuration and status information of the resources. Because of the diversity of the Grid, however, it is difficult to develop an all-encompassing Grid architecture.

Recently, a Grid architecture has been proposed [17] that comprises five layers:

- A ***fabric layer***, which interfaces to local control including physical and logical resources such as systems, files, or even a distributed file system.
- A ***connectivity layer***, which defines core communication and authentication protocols supporting Grid specific network transactions.
- A ***resource layer***, which allows the sharing of a single resource.
- A ***collective layer***, which allows resources to be viewed and operated on as collections.

Copyright © 2001 John Wiley & Sons, Ltd.

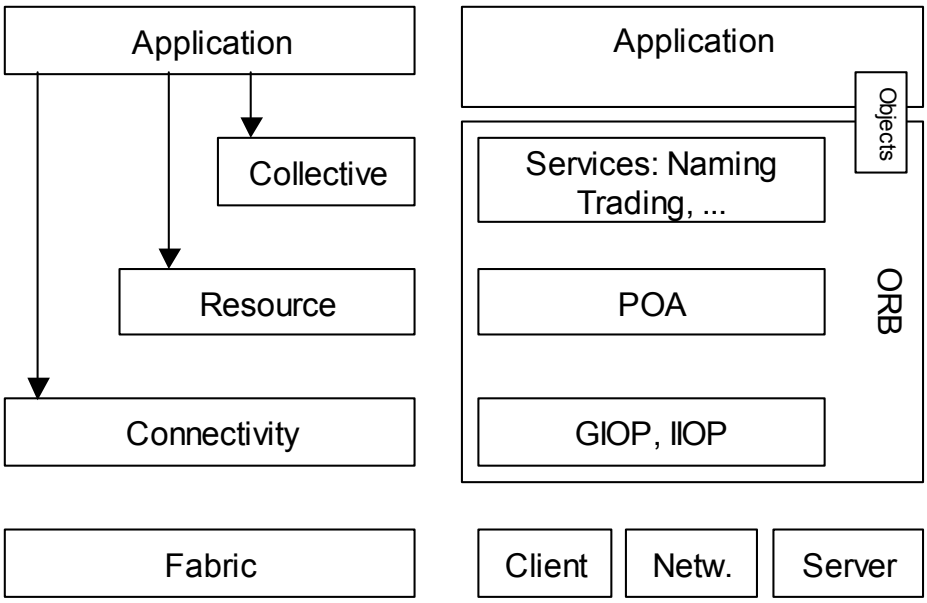

Figure 1: The Grid Architecture and CORBA (The figure on the left shows the Grid architecture. The figure on the right shows how CORBA fits into the Grid Architecture).

Table 1: Mapping of various CORBA-related technologies into the Grid layers.

Collective	CORBA Services: Transaction, Trading Object, Time, Security, Relationship, Query, Property, Persistent Object, Notification, Life Cycle, Licensing, Naming, Externalization, Event, Concurrency, Collection
Resource	POA
Connectivity	GIOP, IIOP, GSI, SSL
Fabric	Client, Server, Networks

Copyright © 2001 John Wiley & Sons, Ltd.

- An ***application layer,*** *which* uses the appropriate components of each layer to support applications.

Each of these layers may contain protocols, APIs, and software development kits (SDKs) to support the development of Grid applications. This general layered architecture of the Grid is shown in the left part of Figure 1.

3. CORBA FOR GRID COMPUTING

Researchers have expressed a growing interest in combining the functionality of Grid technologies and CORBA since CORBA services can support the Grid architecture (as is clear from Figure 1 and Table 1). There are additional reasons why CORBA appeals to users; some of the most important ones are as follows:

- *High-level, modular programming model:* The CORBA interaction model, as well as its services, provides a convenient environment for distributed computation; CORBA hides the complexities of networking and provides security mechanisms and other ready-made solutions. Programming in CORBA not only speeds the development process but also results in systems with a high reusability potential.
- *Interoperability of heterogeneous components:* Components implemented in different languages can interact by specifying interfaces in the (language neutral) Interface Definition Language (IDL).
- *Location transparency*: The CORBA distributed computing model hides the fact that two components may be interacting remotely.
- *Open standard*: CORBA is vendor independent, which results in many implementations on diverse platforms.
- *Interoperability*: CORBA defines mechanisms that allow solutions from different vendors to interoperate.
- *Legacy integration*: Legacy applications can be cast as CORBA objects [26].

The interest in CORBA within the Grid community has led to a number of applications [21, 22, 24, 26] seeking to combine the functionality of CORBA and the Globus Toolkit [5]. Although these solutions work well to solve specific problems encountered in individual applications, they lack generality and uniformity of approach. The

Copyright © 2001 John Wiley & Sons, Ltd.

different CORBA Grid solutions are not necessarily compatible with each other, and they require programmers to frame their solutions in terms of two different programming models that are not always consistent. The purpose of our work is to examine the affinities of these two models, as well as the breadth of functionality they cover, and to define a consistent set of functionality that would fulfil the needs of CORBA Grid applications.

We have identified two key scenarios in which users may want to combine the functionality of the Grid technologies and CORBA:

1. A CORBA programmer may want to combine the CORBA programming model and CORBA services with the functionality provided by the Grid.
2. A Grid programmer may want to access CORBA services not provided by the Grid.

While we plan to address both scenarios, in this paper we focus on providing a high-level CORBA interface to the Grid. We begin by defining a CORBA Grid computing model and mechanisms for making Grid services accessible through the CORBA programming interface.
Since CORBA defines both high-level interoperability (through high-level bridges) and low-level interoperability (through GIOP and IIOP), and acknowledges their respective advantages and disadvantages, we believe that our work on creating interoperability between the Grid and CORBA architectures can benefit from such a dual approach. Therefore, we will pursue two lines of investigation: a high-level approach and a low-level approach. In the former, adding Grid functionality to CORBA is achieved by wrapping CORBA interfaces around key Grid services as described in this paper. The advantages of this approach are simplicity, modularity (i.e., the programmer can use a subset of Grid services and functionality fulfilling the application requirements), and the speed with which a system can be implemented and deployed. The disadvantages are that the approach does not expose all features of CORBA. For example, consider the security service; in this approach the Grid security mechanisms are accessed by using security services present in CORBA, essentially requiring the presence of two largely overlapping security models.
In the low-level approach the CORBA programming model is overlaid on a Grid-based implementation. In this case the CORBA security service (including its interface, if

Copyright © 2001 John Wiley & Sons, Ltd.

necessary) is extended to include Grid-based functionality. Within this approach we plan to experiment with uniting models for specific services rather than presenting them as external components. For example, rather than translating between two different security models, we will consider whether they can interoperate at a lower level, presenting a consistent interface to the user. Similarly, rather than presenting the Grid Resource Allocation Manager (GRAM) [14] as an external service, we will consider how it might fit within the CORBA activation mechanisms. The advantages of this approach are that the programmer deals with one consistent model available through familiar interfaces (CORBA mechanisms). The disadvantages are that this approach is harder to implement and may involve extending many of the CORBA facilities beyond the standard as defined today.

We believe that our final solution will incorporate both approaches to combining CORBA and Grid. The low-level approach will provide the best, and in many cases also the most efficient solution, whenever critical functionality or services present in both Grid and CORBA need to be combined. On the other hand the high-level approach is appropriate when optional Grid-specific functionality (for example, a replica service) needs to be represented. We anticipate that our final solution will contain three kinds of services: pure CORBA services (for example, the persistent state) combined Grid CORBA services (for example, security, object adapters, and other critical services with counterparts in both Grid and CORBA) and pure Grid services (for example, a replica service).

4. ARCHITECTURE

A schematic view of the architecture of the CORBA CoG Kit is shown in Figure 2. The CORBA ORB forms the middle tier, providing clients access to CORBA server objects that implement services on the Grid. Our current implementation includes Grid services provided by the Globus Project [5]. The Globus Toolkit provides an authentication service as part of the Grid Security Infrastructure (GSI) [12, 16, 18], an information service called Metacomputing Directory Service (MDS) [13, 20], a job submission service called Grid Resource Allocation Manager (GRAM) [14], and data storage/access service called Globus Access to Secondary Storage (GASS) [10]. Server objects are wrappers around corresponding Globus services. Clients access these server objects using the CORBA naming service, which maps names to object references. The CORBA security service is used for authenticating clients and enabling them to interact

Copyright © 2001 John Wiley & Sons, Ltd.

securely with server objects. The server objects notify clients of any status changes using the CORBA event service. We are currently expanding the CORBA CoG Kit to provide server objects for other services on the Grid such as DISCOVER [4, 23], or NetSolve [9]. The goal is to provide uniform access to a pool of services that can be used and composed by user applications. The definition of server objects for Globus services is described in the following sections.

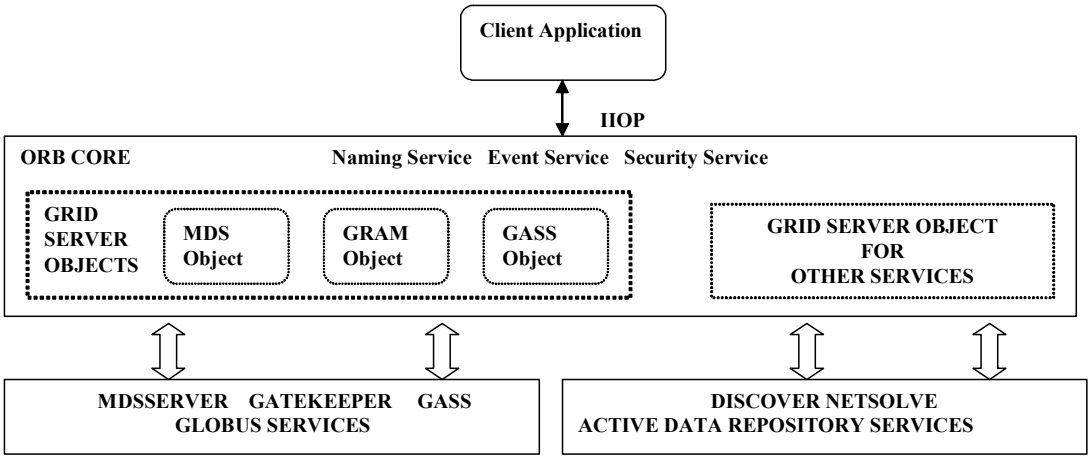

Figure 2: Overall Architecture of CORBA CoG Kit.

5. CORBA INTERFACES TO GLOBUS GRID SERVICES

In what follows we present the interfaces and mechanisms used by the CORBA CoG Kit to provide access to services part of our architecture.

Grid Information Services

To support information services on the Grid, we have decided to develop an interface to the Globus MDS. Although it is possible to develop a COS naming service to access objects stored within the MDS, it is problematic because object definitions in the LDAP (Light Weight Directory Access Protocol) [19] data model are only created at the time

Copyright © 2001 John Wiley & Sons, Ltd.

of instantiation. Thus, it is much easier to provide a direct interface to MDS, returning objects as in the Java CoG Kit [28, 29]. This approach is useful for those familiar with Grid services. CORBA developers may provide their own custom-designed information services based on the CORBA trader interface. We assume that application developers will develop such application specific trader services. In what follows we describe the interface to the MDS and provide an example trader service.

The MDS Server Object

The Metacomputing Directory Service provides the ability to manage and access information about the state of a Grid. It enables read access to entities such as computers, networks, and people. The MDS is based on a distributed directory and is accessed through LDAP [19]. A Grid application can access information about the structure and state of the Grid through the MDS. Information in the MDS is structured using a standard data model consisting of a hierarchy of entities. Each entity is described by a set of "objects" containing typed attribute-value pairs.

The CORBA MDS server object implements a simple interface that provides the following functionality:

1. Establishing connection to the MDS server
2. Querying the MDS server
3. Retrieving results obtained from the MDS query
4. Disconnecting from the MDS server

At the back end, the CORBA MDS server object accesses Globus MDS using JNDI (Java Naming and Directory Interface) [25] libraries, that is, it replicates the approach used by the Java CoG Kit [28]. The IDL we have designed for this purpose is listed in **Figure 3**. The data types returned by the calls to the MDS server are very specific to the JNDI libraries. Since CORBA is a language-independent middleware, it is necessary to map these JNDI specific data types into a generic data type. This mapping is achieved by the structures (viz. *Result, ListResult, MDSList, MDSResult*) defined within the MDS service object. For example, when the *getAttributes()* method is invoked on the CORBA MDS server object, the JNDI libraries return an array of

Copyright © 2001 John Wiley & Sons, Ltd.

```
module MDSService {
    struct Result {
        string id;
         sequence<string> value;
    };

    typedef sequence<Result> MDSResult;
    typedef sequence<string> Attributes;

    struct ListResult {
       string id;
       MDSResult value;
    };

     typedef sequence<ListResult> MDSList;

     interface MDSServer {
        exception MDSException {
           string mdsMessage;
           string ldapMessage;
     };

     void connect (in string name, in long portno, in string username,
                        in string encrypted_password )
                              raises (MDSException);
     void disconnect() raises (MDSException);

     MDSResult getAttributes(in string dn) raises (MDSException);
     MDSResult getSelectedAttributes (in string dn, in Attributes attrs)
                                       raises (MDSException);

     MDSList getList(in string basedn) raises (MDSException);
     MDSList search( in string baseDN, in string filter, in long searchScope)
                                    raises (MDSException);

      MDSList selectedSearch (in string baseDN, in string filter, in Attributes attrToReturn,
                                                    in long searchScope) raises (MDSException);

}
```

Figure 3: The IDL for accessing CORBA MDS Service.

Copyright © 2001 John Wiley & Sons, Ltd.

NamingEnumeration objects that have to be mapped into a customized *Result* data variable. The process involves retrieving the *id* and *attribute* for each *NamingEnumeration* object in this array as string types and then storing the string array as the value variable in the *Result* object. An array of this *Result* object forms the *MDSResult* data variable. Similarly, the *MDSList* data variable is created by mapping the values returned by the *search()* and *getList()* MDS methods.

Grid Domain Trader Services

The CORBA trader service is used to store advertisements of services from remote objects. In the CORBA CoG Kit, we can use the CORBA trader to provide customized information services based on the requirements of the application. For example, consider the creation of a trader that returns information about the number of free nodes in the remote compute resources (Figure 4). Such a trader server has been successfully prototyped and implemented as part of the Numerical Propulsion System Simulation [22]. Traders may provide bridges between different information sources. They form the basis for a more sophisticated information service within the Grid and can be used to integrate various information sources not based on the MDS.

```
GlobusMachineTrader {
  struct MachineType {
     string dn;
     string hn;
     string GlobusContact;
     long   freenodes;
     long   totalnodes;
  };

  typedef sequence<MachineType>  MachineTypeSeq;
  …
  interface GetMachineInfofromMDS {
      void update_seq();
      void initialize_trader();
      void update_trader();
      void refresh_trader();
  };
};
```

Figure 4: A simple example for a CORBA trader service for accessing MDS information.

Copyright © 2001 John Wiley & Sons, Ltd.

Handling Grid Security in CORBA

Providing access to Grid security is an essential part of the CORBA CoG Kit. We base our current implementation on the Globus Grid Security Infrastructure (GSI) [18]. GSI provides protocols for authentication and communication and builds on the Transport Layer Security (TLS) protocols [15]. GSI addresses single sign-on and delegation in virtual organizations. Other features include integration with local security solutions, and user-based trust relations while addressing cross-organizational security issues. While using GSI we are able to use future enhancements provided by the Grid community that deal with multiple CA structures, revocation lists, and community authorization services. One can integrate Grid security at various levels of the CORBA architecture. In order to provide portability across ORBs, we have not considered the modification of the protocol stack. Instead we have placed an intermediary object between the CORBA client and the Grid services. We refer to this intermediary as a CORBA GSIServer object. Authentication is performed using the following three steps:

1. The client and the CORBA server mutually authenticate each other using the CORBA security service (CORBASec) [2, 7, 11]. One of the basic requirements for mutual authentication in CORBASec is to have private credentials, that is, a public certificate signed by a trusted certificate authority (CA), at both the client and server side. In our architecture, both the CORBA client and server use Globus credentials where the trusted certificate authority is accepted by the virtual organization.
2. As Globus services, such as GRAM [14] and GASS [10], only accept connections from clients who have secure Grid credentials (a public certificate signed by a trusted certificate authority), the CORBA client delegates the CORBA GSIServer object to create a secure proxy object that has the authority to communicate with other GSI enabled Grid services on the clients behalf.
3. After successful delegation other Grid service objects may use the secure proxy object to invoke secure connections to the corresponding Globus services to access their functionality.

In more detail, the process of delegation from the client to the CORBA server object involves the following steps. First, the client sends over its public certificate in an encoded form to the server object. Next, the server object generates a completely new

Copyright © 2001 John Wiley & Sons, Ltd.

pair of public and private keys and embeds the public key it received from the client in a certificate request. The certificate request is signed by the new private key and sent across to the client. The client retrieves the public key from the certificate request and from it generates a new certificate. This new certificate, called a proxy certificate, is signed by the client's original private key (not the one from the newly generated pair), and is sent back to the server object in an encoded form. The server object thus creates a chain of certificates where the first certificate is the proxy certificate, followed by the client certificate and then the certificate of the CA. It can then send this certificate chain to the gatekeeper as proof that it has the right to act on behalf on the client. The gatekeeper verifies the chain by walking through it starting with the proxy certificate, searching for trusted certificates and verifying the certificate signatures on its way. If no trusted certificate is found at the base of the chain, the gatekeeper throws a *CertificateException* error. The server object thus uses this certificate chain for establishing a secure socket connection with the gatekeeper. The gatekeeper authenticates the server object on behalf of the client and can accept the clients' request from the server. The IDL interfaces for establishing a secure connection are shown in Figure 5.

```
module GSIService {
  interface GSIServer{
      typedef sequence<octet> ByteSeq;
      void setClientCredentials(in ByteSeq certificate);
      ByteSeq getCertificateRequest();
      void setDelegatedCertificate(in ByteSeq certificate);
   };
};
```

Figure 5: The IDL for accessing the CORBA GSI Service.

The methods in this interface are described below

- setClientCredentials(): This method is called by the client for sending it's public certificate to the server in an encoded form. The client can access this method only after it has securely authenticated with the server.

Copyright © 2001 John Wiley & Sons, Ltd.

- getCertificateRequest(): This method provides the client access to the certificate request generated at the server end.
- setDelegatedCertificate(): Using the certificate request obtained from the server the client generates a new certificate, called the proxy certificate, for delegating the right to access the Globus services to the server. By invoking this method the client can send the proxy certificate in an encoded form to the server.

Job Submission in Grids

The remote job submission capabilities provided by the CORBA CoG Kit uses the Globus GRAM service [14]. Job submission through CORBA GRAM is protected by authentication through GSI as described in the preceding section. The job submission process consists of the following steps: First, the client authenticates with the CORBA service object using CORBASec. After mutual authentication is successful, the client subscribes to the CORBA event channel on which the server is listening. Next the client gets a handle to the GSIServer object using the CORBA naming service, and delegates the CORBA GSIServer object as described in the preceding section. Once delegation is successful, the client obtains a reference to the GRAMServer object (using the CORBA naming service) and submits a Globus job request consisting of the name of the executable, the appropriate parameters, and the name of the resource on which the job is to be executed. On receiving the request, the GRAMServer uses the secure proxy object created by the GSIServer during delegation to initiate a secure socket connection with the GRAM gatekeeper (see Figure 6). It then forwards the request to the gatekeeper and waits for updates on the status of the job via the CORBA event channel. The implementation of the GRAMServer object in the CORBA CoG Kit provides a simple interface with the following methods (see Figure 7):

- jobRequest(): Once the delegation is successfully completed, the client can request job submission on a remote resource by invoking this method on the server.
- setProxyCredentials(): This method sets the reference to the secure proxy object created by the GSIServer object.

Additionally we use the following data structure to monitor the status of the job:

Copyright © 2001 John Wiley & Sons, Ltd.

- JobStatus: It is used by the CORBA event service to notify the client of status changes. The structure consists of two string data types – the *jobid* and the *jobstatus*. The *jobid* identifies the id of the submitted job, and the *jobstatus* contains one of the following values – PENDING, DONE, ACTIVE, FAILED, or SUSPENDED.

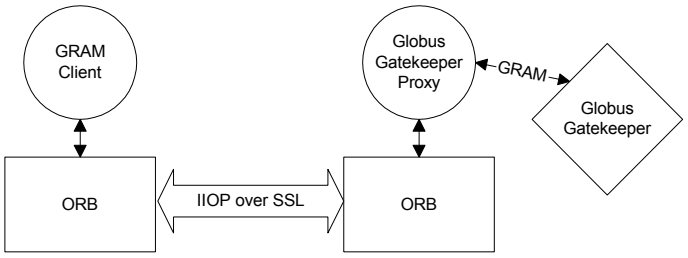

Figure 6: Job submission strategies provided by the CORBA CoG.

```
module GRAMService {
    exception GramException{short errorcode;};

    exception GlobusProxyException{short errorcode;};

    struct JobStatus{
        string jobid;
        string currstatus;
    };

    interface GRAMServer{
       typedef sequence<octet> ByteSeq;
       void setProxytCredentials(in ByteSeq certificate);
       void jobRequest(in string rsl, in string contact,
                       in boolean batchjob);
    };
};
```

Figure 7: The IDL for accessing CORBA GRAM Service.

Copyright © 2001 John Wiley & Sons, Ltd.

Data Transfer on the Grid

A frequent problem that needs to be addressed in Grid applications is to access remote data - for example, when the application wants to pre-stage data on remote machines, cache data, log remote application output in real time or stage executables on a remote computer. In our current implementation we use the Globus GASS service for data transfer between resources in Grids. The goal of GASS is not to build a general-purpose distributed file system but to support I/O operations commonly required by Grid applications. The strategy employed is to fetch the file and cache it on first read open, and write it to disk when it is closed.

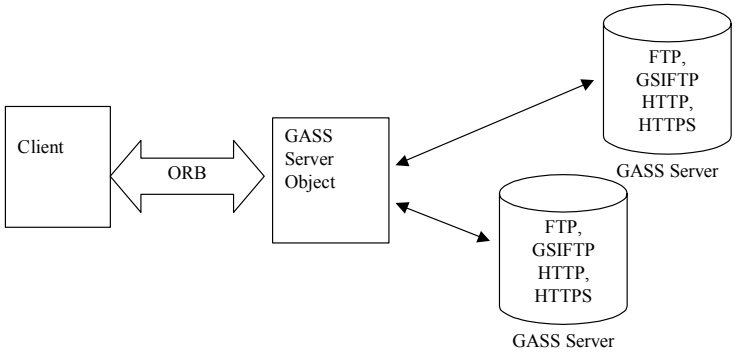

Figure 8: The CORBA CoG Kit interface to GASS.

The objective of the CORBA GASSServer object is to provide an interface to the Globus GASS service as shown Figure 8. The client gets a handle to the GASSServer object from the CORBA naming service, and this server object forwards the client's requests to the appropriate GASS servers using the protocol specified by the client. GASS supports FTP, HTTP, HTTPS, and GSIFTP. Both the FTP and GSIFTP protocols allow third-party file transfers; that is they allow file transfers from a sender machine to a receiver machine to be initiated by an third initiator machine. Both the sender and receiver machines have to provide a GASS server. Authentication is performed using GSI. The methods defined by the CORBA GASS service interface are shown in Figure 9.

Copyright © 2001 John Wiley & Sons, Ltd.

```
module GASSService {
  interface GASSServer {
      void setProxyCredentials();
      void setSourceURL(in string sourceurl);
      void setDestinationURL(in string destnurl);
      void allowThirdPartyTransfer(in boolean value);
      void URLcopy()
  };
};
```

Figure 9: The IDL of the CORBA GASS Service.

6. APPLICATION OF THE CORBA CoG KIT

Many applications can benefit from a CORBA CoG Kit. One example is the Numerical Propulsion System Simulation (NPSS), which is a part of the NASA IPG (Information Power Grid) and provides an engine simulation using computational fluid dynamics. It consists of 0-D to 3-D engine component models responsible for examining aerodynamics, structures, and heat transfer. Previous studies show that the NPSS engine components can be encapsulated using CORBA to provide object access and communication from heterogeneous platforms, while at the same time coordinating the modelling runs across the Grid. In this application, a large number of NPSS jobs (1000+) are submitted, using the CORBA CoG Kit, from a desktop interface and return their outputs to that same interface. The engine model for these runs consists of 0-D and 1-D CORBA wrapped components that can be executed quickly. To execute these components using the CORBA CoG Kit, the application initially discovers the resources available on the NASA Grid by executing a query on CORBA MDS service. The service connects to the NASA MDS server and retrieves the list of machines that satisfy the structure and state of the query. From this list of available machines, the application selects a particular machine or a cluster of machines and prepares it for executing the job. Since the simulation output must be transferred back to the desktop interface, the output file has to be initially copied to the execution machine using CORBA GASS service. This service can also be used for staging the application. Once the files are transferred, the CORBA GRAM service is used for submitting the job on the executing machine. Thus, by using the three basic services provided by the CORBA CoG Kit, jobs can be executed, without requiring manual initialisation of the ORBs, starting of services, and transferring of data files to the remote systems. The pseudocode presented in Figure 10 illustrates how the CORBA Grid services are accessed and the engine component model is executed on a remote machine.

Copyright © 2001 John Wiley & Sons, Ltd.

```
//Initialization of the orb
orb = org.omg.CORBA.ORB.init(args, props);

//After authentication of client and the server ets the reference to the event channel for communication between the client and the server
org.omg.CORBA.Object obj = orb.resolve_initial_references("EventChannelFactory");
EventChannelFactory f = com.ooc.OBEventChannelFactory.EventChannelFactoryHelper.narrow(obj);
EventChannel e = f.get_channel_by_id("GlobusEventChannel");

// Get a reference to Naming Service
obj = orb.resolve_initial_references("NameService");

// The naming service has a NamingContext named Globus which has references to all the CORBA Globus Services
org.omg.CORBA.Object nc1Obj = nc.resolve("Globus");

// Reference to GRAMService is obtained
org.omg.CORBA.Object implobj = globusctx.resolve("GramService");
GRAMServer gramserver = GRAMServerHelper.narrow(implobj);

// Reference to MDSService is obtained
org.omg.CORBA.Object implobj = globusctx.resolve("MDSService");
MDSServer mdsserver = MDSServerHelper.narrow(implobj);

//Reference to GASSService is obtained
org.omg.CORBA.Object implobj = globusctx.resolve("GASSService");
GASSServer gassserver = GassServerHelper.narrow(implobj);

// Connect to the MDSService and get a list of available machines
mdsserver.connect("mds.as.nren.nasa.gov",389,"", "");
Result[] result;
java.lang.String strDN = "ou=Glenn Research Center, o=National Aeronautics and Space Administration, o=Grid";
result = mdsserver.getAttributes(strDN);

//Search for an machines having particular structure
ListResult[] listresult=mdsserver.search("ou=Glenn Research Center, o=National Aeronautics and Space Administration, o=Grid
","&((object-class=GridComputeResource)(freenodes=64))", "",  MDS.ONELEVEL_SCOPE );

//Prepare the data for experiment
// Transfer the output file from the desktop to the remote machine
gassserver.setSourceURL("file://e:/output");
gassserver. setDestinationURL(https://nasa.edu/~david/output);
gassserver.URLcopy();

// Delegate the server
gramserver.setClientCredentials(public_certificate.getEncoded());
byte[] newcertreq = gramserver.getCertificateRequest();
ByteArrayInputStream bis = new ByteArrayInputStream(newcertreq);
iaik.pkcs.pkcs10.CertificateRequest req = new iaik.pkcs.pkcs10.CertificateRequest(bis);
java.security.cert.X509Certificate proxy_certificate = proxy_sign(req);
gramserver.setDelegatedCertificate(proxy_certificate.getEncoded());

//Job Submission
String rsl = "&(executable='enginemodel')(processors=64)"
String contact = listresult[0].id

//Submits the job and gets an update of the status of the job through the event channel
gramserver.jobRequest(rsl, contact , false);
orb.run()
```

Figure 10: Pseudocode showing the use of CORBA CoG Services.

Copyright © 2001 John Wiley & Sons, Ltd.

This application can be also integrated with other services such as DISCOVER [4, 23]. DISCOVER allow users to collaboratively monitor and control application, access, interact, and steer individual component objects; manage object dynamics and distribution; and schedule automated periodic interactions. The 0-D or 1-D engine components that are submitted for execution via the CORBA GRAM server can be interactively steered and collaboratively monitored using DISCOVER. Other examples include the control of advanced scientific instruments such as radio telescopes and synchrotron rings, via their commercially available control infrastructure, using access through CORBA objects. At many of these installations it will not be possible to install Globus server side software but only to interface to it as a client.

7. STATUS

The current implementation of the CORBA CoG Kit provides MDS, GSI, GRAM, and GASS services. In the case of GIS, we have concentrated on a direct interface to the MDS. Previous effort has demonstrated that it is straightforward to generate specialized trading services in CORBA. Once we have identified a suitable set of requirements from different applications, we will develop for these application domain specific trading services. The performance of the CORBA GRAM service with different ORBs is currently being evaluated. We have also made significant progress in integrating the CORBA CoG Kit with DISCOVER. The current status of the CORBA CoG Kit project as well as the software can be obtained from www.caip.rutgers.edu/TASSL/Projects/CorbaCoG/. Future activities will include investigations of scalability, the interplay of the integration between CORBA and Grid technologies, and comparisons for information exchange, either through massage services or through remote object access.

8. CONCLUSION

This paper reports on an ongoing project aimed at designing, implementing and deploying a CORBA CoG Kit. The overall goal of this project is to provide a framework that will enable existing Grid computing environments and CORBA service providers to interoperate. CORBA is an accepted technology for building distributed applications and is widely used and supported by academia and industry. Its features include a high-level modular programming model, availability of advanced services

Copyright © 2001 John Wiley & Sons, Ltd.

(e.g. security, naming, trading, event, transaction, etc.) and readymade solutions, interoperability, language independence, location transparency and an open standard, making it a suitable candidate for developing Grid applications. Developing a CORBA CoG Kit facilitates this integration. The demand for such a CoG Kit has been expressed by various projects ranging from the creation of CORBA based control systems for advanced instruments to the collaborative interaction and computational steering of very large numerical relativity and fluid dynamics applications. Our current efforts are focused on enabling applications to combine and compose services on the Grid – e.g. combining services provided by Globus, with the collaborative monitoring, interaction, and steering capabilities provided by DISCOVER [4, 23]. For example a scientific application can use CORBA CoG Kit to discover the available resources on the network, use the GRAM Service provided by CoG to run the simulation on the desired high end resource, and use DISCOVER web portals to collaboratively monitor, interact with, and steering the application.

ACKNOWLEDGMENTS

We thank Brian Ginsburg, Olle Larsson, Stuart Martin, Steven Tuecke, David Woodford, Isaac Lopez, Gregory J. Follen, Richard Gutierrez, and Robert Griffin for their efforts to provide a C++ -based CORBA interface for the NPSS application performed at NASA Glenn Research Centre. We also thank Viraj Bhat for his contributions to the CORBA-CoG project.

This work was supported in part by the National Science Foundation under Grant Number ACI 9984357 (CAREERS), awarded to Manish Parashar; by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, U.S. Department of Energy, under Contract W-31-109-Eng-38; by the Defence Advanced Research Projects Agency under contract N66001-96-C-8523; and by the NASA Information Power Grid program.

APPENDIX

Table A-1summarizes an evaluation of available ORBs. The evaluation is based on some basic features relevant in the development of the CORBA CoG Kit. These include OMG CORBA standard compliance, services provided, languages and platforms supported, and cost [1].

Copyright © 2001 John Wiley & Sons, Ltd.

Table A-1: Comparison of available ORBs, where $ indicates commercial license.

ORB	OMG Compliance	Services Available	OS	Prog. Languages	POA	Compatibility	Cost
Orbix 2000	CORBA 2.3	All major services available	Win, Solaris, Linux, HPUX, AIX, Compaq	C++ & JAVA	Yes		$
OmniORB 3.0.3	CORBA 2.3	No Interface Repository	Win, Solaris, Linux, HPUX, SGI IRIX	C++	Yes	IONA orbix, Visibroker, HP ORB Plus	Free
Visibroker 4.5	CORBA 2.3	All major services. For security service it requires Borland security service	Win, Solaris, Red Hat Linux, HP-UX, IBM AIX	Java, C++	Yes		$
Orbacus 4	CORBA 2.3	Naming, Event, Property & Time Service	Win, Solaris, Red Hat Linux, HP-UX, IBM AIX	Java, C++	Yes		$
JacORB 1.3.21	CORBA 2.3	All major services. For security service it mentions improved IIOP over SSL	Linux, Win, Unix	Java	Yes	MICO, TAO, Orbacus, OrbixWeb, VisiBroker, ORBit, omniORB, Vitria C++ and Java ORBs	Free
ACE TAO	CORBA 2.4	All major services including portions of aaecurity service	Unix, Win, Linux	C++	Yes	Visibroker, Orbix, JacORB, ORB Express, VisiBroker	Free
MICO 2.3.5	CORBA 2.3	All major services. Security service is under development	Solaris, IBM AIX, HP-UX, Linux, Digital Unix, Ultrix Win	C++	Yes		Free
MICOSec	Based on Mico	Security service Level2 Ver 1.7	Solaris, Win, Linux	C++			$
Fnorb 1.1	CORBA 2.0	Interface Repository, Naming Service	Unix, Win95, NT	Python	No		Free for non-$ use

Copyright © 2001 John Wiley & Sons, Ltd.

ORBit 0.5.8	CORBA 2.2	All major services including security service	Unix, Linux, Win	Mainly for C & Perl. Others are C++, Lisp, Pascal, Python, Ruby, TCL	Yes		Free

REFERENCES

1. Adrion Secure System Design, Web Page, 2001, http://www.adrion.com.
2. CORBA Security Service Specification, 2001, ftp://ftp.omg.org/pub/docs/formal/01-03-08.pdf.
3. The DISCOVER Project, 2001, http://www.caip.rutgers.edu/TASSL/Projects/DISCOVER/Documents.html.
4. V. Mann and M. Parashar, Middleware Support for Global Access to Integrated Computational Collaboratories , *Proc 10th IEEE International Symposium on High Performance Distributed Computing,*(2001), 35-46.
5. Globus Web Page, 2001, http://www.globus.org.
6. The Grid Portal Development Kit, 2001, http://dast.nlanr.net/Features/GridPortal/.
7. OMG Security, 2001, http://www.omg.org/technology/documents/formal/omg_security.htm.
8. The Python CoG Kit, 2001, http://www.globus.org/cog.
9. D. C. Arnold and J. Dongarra, The NetSolve Environment: Progressing Towards the Seamless Grid. *Proc. International Workshop on Parallel Processing*, (2000).
10. J. Bester, I. Foster, C. Kesselman, J. Tedesco and S. Tuecke, GASS: A Data Movement and Access Service for Wide Area Computing Systems. *Proc. IOPADS'99*, (1999).
11. B. Blakley, R. Blakley and M. Soley, CORBA Security: An Introduction to Safe Computing with Objects. *The Object Technology Series.* Addison-Wesley,
12. R. Butler, D. Engert, I. Foster, C. Kesselman, S. Tuecke, J. Volmer and V. Welch, Design and Deployment of a National-Scale Authentication Infrastructure. *IEEE Computer*, (1999), 33 (12). 60-66.
13. K. Czajkowski, S. Fitzgerald, I. Foster and C. Kesselman, Grid Information Services for Distributed Resource Sharing. *Proc. 10th IEEE International*

Copyright © 2001 John Wiley & Sons, Ltd.

Symposium on High Performance Distributed Computing, (2001), http://www.globus.org.

14. K. Czajkowski, I. Foster, N. Karonis, C. Kesselman, S. Martin, W. Smith and S.Tuecke, A Resource Management Architecture for Metacomputing Systems. *Proc. IPPS/SPDP '98 Workshop on Job Scheduling Strategies for Parallel Processing*, (1998), 62-82.
15. T. Dierks and C. Allen, The TLS Protocol Version 1.0, IETF, (1999), http://www.ietf.org/rfc/rfc2246.txt.
16. I. Foster, N. T. Karonis, C. Kesselman and S. Tuecke, Managing Security in High-Performance Distributed Computing. *Cluster Computing*, (1998), 1(1), 95-107.
17. I. Foster, C. Kesselman and S. Tuecke, The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International J. Supercomputer Applications*, (2001), 15(3).
18. I. Foster, C. Kesselman, G. Tsudik and S. Tuecke, A Security Architecture for Computational Grids. *Proc. 5th ACM Conference on Computer and Communications Security Conference*, (1998), 83-92.
19. T. Howes, M. C. Smith, G. S. Good and T. A. Howes, *Understanding and Deploying Ldap Directory Services*. Mac Millan, 1999, LDAPbook.
20. G. v. Laszewski, S. Fitzgerald, I. Foster, C. Kesselman, W. Smith and S. Tuecke, A Directory Service for Configuring High-Performance Distributed Computations. *Proc. 6th IEEE Symposium. on High-Performance Distributed Computing*, (1997), 365-375.
21. The Advanced Computational Concepts Laboratory of the NASA Glenn Research Center, Web Page, 2001, http://accl.lerc.nasa.gov/IPG/CORBA/.
22. I. Lopez, G. J. Follen, R. Gutierrez, I. Foster, B. Ginsburg, O. Larsson and S. Tuecke, Using CORBA and Globus to Coordinate Multidisciplinary Aeroscience Applications. *Proc. NASA HPCC/CAS Workshop*, (2000), http://accl.lerc.nasa.gov/IPG/CORBA/NPSS_CAS_paper.html.
23. 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*, (2001), 13(8-9), 737-754.
24. H. Prot, M. Bouet, V. Breton, S. Du, N. Jacq, Y. Legre, R. Medina, R. Metery and J. Montagnat, A Virtual Laboratory for Bioinformatics on the GRID, European Community document, (2001).

Copyright © 2001 John Wiley & Sons, Ltd.

25. S. S. Rosanna Lee, *JNDI API Tutorial and Reference: Building Directory-Enabled Java Applications*. Addison-Wesley, 2000.
26. J. Sang, C. Kim and I. Lopez, Developing CORBA based distributed scientific applications from legacy FORTRAN applications. *Proc. of HPCC Computational Aerosciences (CAS)*, (2000).
27. S. Verma, M. Parashar, J. Gawor and G. von Laszewski, Design and Implementation of a CORBA Commodity Grid Kit. *Second International Workshop on Grid Computing - GRID 2001*, Spinger LNCS 2242, Denver, (2001), 2-12, http://www.caip.rutgers.edu/TASSL/Papers/corbacog-gcw01.pdf.
28. G. von Laszewski, I. Foster, J. Gawor and P. Lane, A Java Commodity Grid Kit. *Concurrency and Computation: Practice and Experience*, (2001), 13 (8-9). 643-662, http://www.globus.org/cog/documentation/papers/cog-cpe-final.pdf.
29. G. von Laszewski, I. Foster, J. Gawor, W. Smith and S. Tuecke, CoG Kits: A Bridge between Commodity Distributed Computing and High-Performance Grids. Proc. *ACM 2000 Java Grande Conference*, San Francisco, CA, (2000), 97-106, http://www.mcs.anl.gov/~laszewsk/papers/cog-final.pdf.
30. Y. Wang, F. D. Carlo, D. Mancini, I. McNulty, B. Tieman, J. Bresnahan, I. Foster, J. Insley, P. Lane, G. v. Laszewski, C. Kesselman, M.-H. Su and M. Thiebaux, A High-Throughput X-Ray Microtomography system at the Advanced Photon Source. *Review of Scientific Instruments*, 72 (4), 2062-2068.

Copyright © 2001 John Wiley & Sons, Ltd.