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,
[1]*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.*
.
Corresponding Author: Gregor von Laszewski, gregor@mcs.anl.gov, phone 630 252 0472, fax 630 252 5986

Submitted to:

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 this software engineering task. 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 support the development and deployment of CORBA applications on the Grid by providing them access to the Grid services provided by the Globus Toolkit.

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 centres, 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 they are often incompatible with 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 **C**ommon **O**bject **R**equest **B**roker **A**rchitecture (CORBA) [5] defined by the Object Management Group (OMG). OMG is an independent consortium of vendors; consequently, the standard it defined is open (vendor independent) and has resulted in many independent implementations for a variety of platforms. In order to provide interoperability between these diverse implementations, CORBA defines interoperability mechanisms. A high-level, distributed computing model, vendor independence, and a strong interoperability thrust all combined to make CORBA an attractive and popular distributed computing standard. As such CORBA meets the necessary requirements to be seriously 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 commodity technology. Currently, CoG Kits are being developed for the Java platform [39] [40], Java Server Pages [11], Python [20], and perl. This paper will describe our experiments in defining yet another CoG Kit: one allowing CORBA applications to access (and provide) services on the Grid. Such integration would provide a powerful application development environment for high-end users and would create a CORBA "Grid domain."

First, we 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. We then briefly outline 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 last section, we identify areas for further activities.

2.   THE GRID

The term "Grid" 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 [30]. 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 center in which a client submits jobs to the supercomputer batch queue. An example for peer-to-peer computing is the collaborative online steering of high-end applications, as demonstrated by the use of advanced instruments [41] [7] [35].

Grids must support different levels of control, ranging from fine-grained access control to delegation, single user to multi-user and collaborations, and different services such as scheduling, coallocation, and accounting. These requirements are not sufficiently addressed by the current commodity technologies, including CORBA. Although sharing of information and communication between resources is allowed, it is not easy to coordinate use of resources at multiple sites for computation. To date the Grid Community has developed protocols, services, and tools that address the issues arising from sharing resources, service, and applications 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 [41] 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.
- 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 (SDK)s to support the development of Grid applications. This general layered architecture of the Grid is shown in the left part of Figure 1.
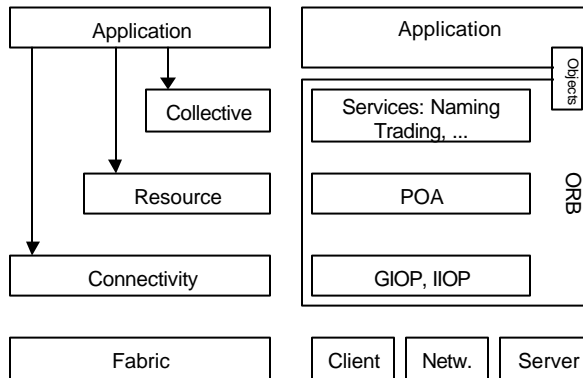

Figure 1: The figure on the left depicts the Grid architecture. The figure on the right depicts how CORBA fits within this architecture.

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

3.   CORBA FOR GRID COMPUTING

Researchers have expressed a growing interest in combining the functionality of Grid technologies and CORBA as CORBA services can support the Grid architecture as is obvious 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 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 over many diverse platforms.
- *Interoperability*: CORBA defines mechanisms that allow solutions from different vendors to interoperate.
- *Legacy integration*: Legacy applications can be cast as CORBA objects [38].

The interest in CORBA within the Grid community has led to a number of applications seeking [14, 34, 36, 38] to combine the functionality of CORBA and Globus [10]. Although these solutions work well to solve specific problems encountered in individual applications, they lack generality and uniformity of approach. The 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 eventually plan to address both scenarios in this paper we will focus on
providing a high-level CORBA interface to the Grid. We will begin by defining a
CORBA Grid computing model and making Grid services accessible through the
CORBA programming interface.

As CORBA defines both high-level interoperability (through high-level bridges) and
low-level interoperability (through IIOP) we pursue both 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 is not exposing all features of
CORBA. For example, let us consider the security service; in this approach the
mechanisms for Grid security are implemented using security services present in
CORBA and the security model of GSI, 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
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 translate between two different
security models we will consider if they can interoperate at a lower level presenting
consistent interface to the user. Similarly, rather than present GRAM [27] as an external
service, we will consider how it might fit within CORBA activation mechanisms. The
advantages of this approach are that the programmer would deal 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, the security, as well as ORB functionality, 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 [10]. Globus provides an authentication service as part of the Grid Security Infrastructure (GSI) [25, 28, 29], an information service called Metacomputing Directory Service (MDS) [26, 32], a job submission service called Grid Resource Allocation Manager (GRAM) [27], and data storage/access service called Globus Access to Secondary Storage (GASS) [23].
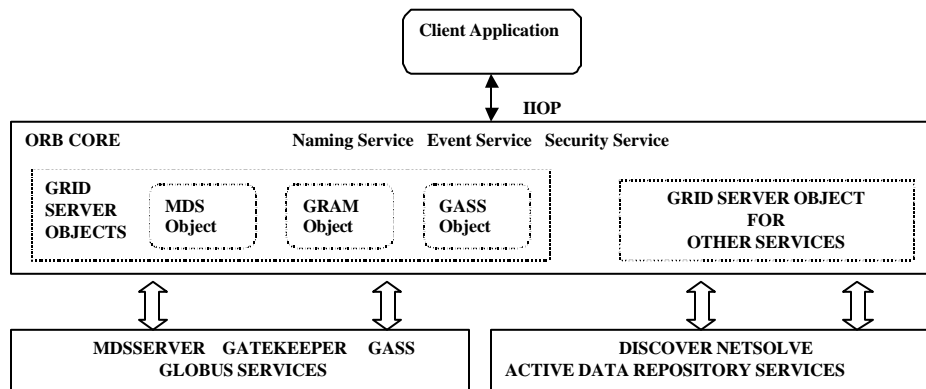

Figure 2: **Overall architecture of CORBA CoG Kit.**

We developed a set of server objects that interact implicitly with the appropriate Grid service. Clients access these server objects using the CORBA naming service, which maps names to object references. The CORBA security service is used for authenticating the clients, and for enabling them to interact securely with server objects. The server objects notify clients of any status changes through 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 [7, 35], or NetSolve [22]. The goal is to provide uniform access to a pool of services that can be used and composed by the user applications. The definition of server objects for the Globus services is described in the following sections.

5.   CORBA INTERFACES TO GLOBUS GRID SERVICES

In what follows we briefly describe the overall architecture and then present the interfaces and mechanisms used by the CORBA CoG Kit to provide access to Grid services as part of our architecture.

Information Services

In order to support information services in the Grid we have decided to develop an interface to the MDS. Although it is possible to develop a COS naming service to access objects stored within the MDS, it is problematic as object definition in the LDAP data model are created at time of instantiation. Thus, it is much easier to provide a direct interface to the MDS, returning objects in the same fashion as the Java CoG Kit. 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 application specific trader services. Next we describe the interface to the MDS and provide an example trader service.

The MDS Server Object

The Metacomputing Directory Service (MDS) 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

```
module MDS {
  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: IDL for the MDS Server object.

accessed through LDAP [31]. 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 construction of the MDS server object is described below.

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 backend, the CORBA MDS server object accesses Globus MDS using JNDI (Java Naming and Directory Interface) [37] libraries, i.e. it replicates the approach used by the Java CoG Kit [39]. 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 specific data types into a generic data type. This is achieved by the structures (i.e. *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 returns an array of *NamingEnumeration* objects that have to be mapped into a customized *Result* data variable. This is done by retrieving the *id* and *attribute* for each *NamingEnumeration* object in this array as string types, and storing the string array as the value variable in the *Result* object. An array of this *Result* object forms the *MDSResult* data variable. Similarly, *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

of remote compute resources (Figure 4). Such a trader server has been successfully prototyped and implemented as part of the Numerical Propulsion System Simulation [33]. Traders may provide bridges between different information sources and 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.

Handling Grid security in CORBA

To provide portability between the Grid and CORBA, authentication is handled by introducing an intermediary between the CORBA client and the Globus GRAM service. We will refer to this intermediary as a "CORBA server object". Authentication is performed with the following three steps:

1. The client and the CORBA server mutually authenticate each other using the CORBA security service (CORBASec) [4, 15, 24].

2. A gatekeeper (part of the GRAM service [27]) accepts only connections from clients who have secure Grid credentials, i.e. a public certificate signed by a trusted certificate authority (CA). The CORBA client delegates the CORBA server object the authority to communicate with the gatekeeper on its behalf using the GSI proxy mechanism.

3. After successful delegation the CORBA server object sets up a secure connection with the gatekeeper.

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 pair of public and private keys and embeds the newly generated public key 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 is called a proxy certificate. It 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' job submission request from the server. The IDL interfaces for establishing a secure connection is shown in Figure 5.

Remote Job Submission on the Grid

The remote job submission capabilities provided by the CORBA CoG Kit uses the Globus GRAM service [27]. Job submission through CORBA GRAM is protected by authentication through GSI as described in the previous section. During the job submission following steps are conducted:

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

     exception GlobusProxyException{short errorcode;};

     struct JobStatus{
         string jobid;
         string currstatus;
};
interface GRAMServer {
     typedef sequence<octet> ByteSeq;
     void setClientCredentials(in ByteSeq certificate);
     ByteSeq getCertificateRequest();
     void setDelegatedCertificate(in ByteSeq certificate);
     void jobRequest(in string rsl, in string contact,
                          in boolean batchjob);};
};
```
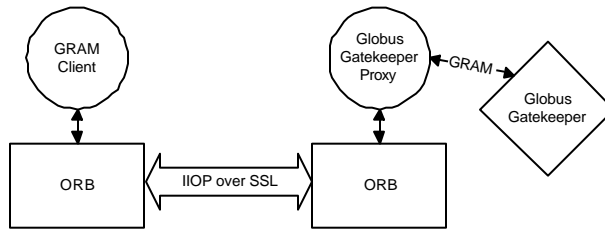
Figure 5: The IDL for accessing CORBA GRAM Service

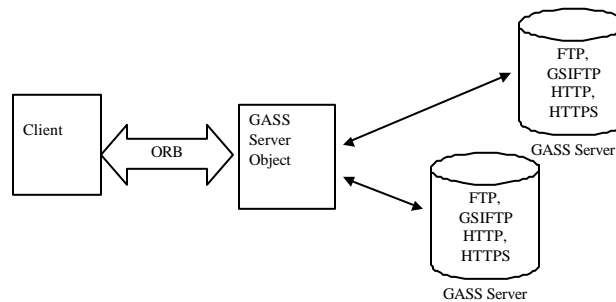**Figure 6:** Job submission strategies provided by the CORBA CoG.

Figure 7: The CORBA CoG Kit interface to GASS.

First, our service authenticates with the CORBA Service object (Figure 8) using CORBASec. After the mutual authentication is successful, the client subscribes to the CORBA event channel on which the server is listening. Next the client delegates the CORBA Server object using the process as described in the previous section. Once the delegation is successful, the client performs a request for job submission by specifying the executable name and the resource name on which the job has to be executed. On receiving the request, the server sets up a secure socket connection with the GRAM gatekeeper. Then, it forwards the request to the gatekeeper and waits for status updates of the job via the CORBA event channel (Figure 6).

The implementation of the CORBA GRAM service object in the CORBA CoG Kit provides a simple interface with the following methods:

- 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.
- getCertificateRequest(): This method provides the client access to the certificate request generated at the server end.
- setDelegatedCertificate(): From the certificate request obtained from the server the client generates a new certificate called the proxy certificate for delegating to the server the right to access the Globus services. By invoking this method the client can send the proxy certificate in an encoded form to the server.
- jobRequest(): Once the delegation is successfully completed the client can request a job submission on a remote resource by invoking this method on the server.

Additionally we use the following data structure to monitor state changes:

- 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.

Data Transfer on the Grid

A frequent problem that needs to be addressed in high performance computations is to access remote data to perform a computation -- for example, when the application wants to prestage data on remote machines, cache data, log remote application output in real time or stage executables on a remote computer. In our first prototype we use Globus GASS [9] for data transfer between resources in Grids. The goal of GASS is not to build a general-purpose distributed file system but to support the I/O operations commonly found in Grid applications. The strategy employed is to fetch the file and cache it on first read open, or write it to disk when it is closed.

The objective of the CORBA GASS service is to provide an interface to the Globus GASS service as shown in Figure 7.. The client connects to a CORBA server, and the server forwards its request to the appropriate GASS servers using the protocol specified by the user. GASS supports the FTP, HTTP, HTTPS, and GSIFTP.

Both the FTP and GSIFTP protocol allows third-party file transfers; that is they allow files to be transferred from a sender machine to a receiver machine, while being initiated by an initiator machine. Sender and receiver machines need to provide a GASS server. Authentication is performed via GSI. The methods defined by the CORBA GASS service interface are shown in Figure 88.

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

Figure 8: The IDL of the CORBA GASS server object.

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) [14], which is a part of the NASA IPG and provides an engine simulation using computational fluid dynamics. It consists of 0- to 3-dimensional engine component models responsible for examining aerodynamics, structures, and heat transfer. Previous studies show that the NPSS's engine

components can be encapsulated using CORBA in order to provide object access and communication from heterogeneous platforms while at the same time coordinating the modelling runs across Globus. As part of this task, a large number of NPSS jobs (1000+) are submitted from a desktop interface returning the output to that same interface using the CORBA CoG Kit. 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 should initially discover 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 can select a particular machine or a cluster of machines and prepare it for running the job. Since the simulation output needs to 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 could also be used for application execution staging. Once the files are transferred the CORBA GRAM service is used for submitting the job on the machines. By using the three basic services provided by the CORBA CoG Kit the jobs can be executed, foregoing the steps of manually initialising ORBs, starting services, and placing data files on remote systems. The pseudocode presented in Figure 9 illustrates how the CORBA Grid services are accessed and the engine component model is executed on a particular machine.

This application can be also integrated with other services such as DISCOVER [35]. DISCOVER service 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.


```
//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 9: Pseudocode for implementing the CORBA CoG Services in an application.

7. STATUS

The current implementation of the CORBA CoG Kit provides MDS, GIS and GRAM 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 by other applications, we will develop for these applications domain specific trading services. The performance of the CORBA GRAM service with different ORBs is being evaluated. We are in the development stage of a CORBA GASS Service. Moreover we have 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 [3].

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 to enable existing Grid computing environments and CORBA service providers to interoperate. CORBA targets distributed environment, is supported by many vendors, and provides transparency on many levels including languages, operating systems, networks, and protocols. It is an ideal candidate for applications programmers to develop Grid-based applications. Providing a CORBA Grid domain will allow an easy integration of additional Grid services and functionality within these applications. 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 computational steering of fluid dynamics codes. Our future effort will concentrate on enabling applications to combine services developed by the Globus project, with the collaborative monitoring, interaction, and steering capabilities distributed with DISCOVER [6]. Our objective is to enable a scientific simulation application using the CORBA CoG Kit to discover the available resources on the network, use the GRAM service provided to run simulations on the desired high-end resources; and use DISCOVER Web-portals to collaboratively monitor, interact with, and steer the application.

ACKNOWLEDGMENT

Appendix

Table 2 summarizes an evaluation of available ORBs. The evaluation based on some basic features relevant in the development of the CORBA CoG Kit. These include OMG CORBA standard compliance, services provided (specially security service), languages and platforms supported, and cost [1 mico, 2, 8, 12, 13, 16-19, 21]

.

Table 2: 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, portions of acecurity ser.	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
ORBit 0.5.8	CORBA 2.2	All major services including security service	Unix, Linux, Win	C & Perl. But also C++, Lisp, Pascal, Python, Ruby, and TCL	Yes		Free

REFERENCES

[1]   "ACETao," 2001, http://www.cs.wustl.edu/~schmidt/TAO.html.

[2]   "Adrion Secure System Design, Web Page," 2001, http://www.adrion.com.

[3]   "The CORBA CoG Kit Source Code,"
        http://www.caip.rutgers.edu/TASSL/CorbaCoG/CORBACog.htm.

[4]   "CORBA Security Service Specification," March ed, 2001,
        ftp://ftp.omg.org/pub/docs/formal/01-03-08.pdf.

[5]   "CORBA: Common Object Request Broker Architecture," http://www.omg.org.

[6]   "The DISCOVER Project," 2001,
        http://www.caip.rutgers.edu/TASSL/Projects/DISCOVER/Documents.html.

[7]   "The DISCOVER Project Web Page," 2001,
        http://www.caip.rutgers.edu/TASSL/Projects/DISCOVER/Documents.html.

[8]   "Fnorb," http://www.fnorb.org/.

[9]   "The Globus GASS Web Page," http://www.globus.org/gass.

[10]   "Globus Web Page," 2001, http://www.globus.org.

[11]   "The Grid Portal Development Kit," 2001,
        http://dast.nlanr.net/Features/GridPortal/.

[12]   "JacORB," http://www.jacorb.org.

[13]   "MICOSec," http://www.objectsecurity.com/micosec.org/.

[14]   "Numerical Propulsion System Simulation (NPSS),"
        http://hpcc.lerc.nasa.gov/npssintro.shtml.

[15]   "OMG Security," 2001,
        http://www.omg.org/technology/documents/formal/omg_security.htm.

[16]   "OmniORB," http://www.uk.research.att.com/omniORB/omniORB.html.

[17]   "ORBacus," www.ooc.com/ob/.

[18]   "ORBit," http://www.labs.redhat.com/orbit/.

[19]   "ORBIX," http://www.iona.com.

[20]   "The Python CoG Kit," 2001, http://www.globus.org/cog.

[21]   "Visibroker," http://www.visibroker.com.

[22]   D. C. Arnold and J. Dongarra, "The NetSolve Environment: Progressing
        Towards the Seamless Grid," presented at Proc. International Workshop on
        Parallel Processing, 2000.

[23]   J. Bester, I. Foster, C. Kesselman, J. Tedesco, and S. Tuecke, "GASS: A Data
        Movement and Access Service for Wide Area Computing Systems," in *Proc.\
        IOPADS'99*, 1999.

[24]   B. Blakley, R. Blakley, and M. Soley, *CORBA Security: An Introduction to Safe Computing With Objects*
," The Object Technology Series.: Addison -Wesley.
[25]   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*, vol. 33, pp. 60-66, 2000.
[26]   K. Czajkowski, S. Fitzgerald, I. Foster, and C. Kesselman, "Grid Information Services for Distributed Resource Sharing," presented at Proc. 10th IEEE International Symposium on High Performance Distributed Computing, 2001.
[27]   K. Czajkowski, I. Foster, and C. Kesselman, "Co-allocation Services for Computational Grids," presented at Proc. 8th IEEE Symposium on High Performance Distributed Computing, 1999.
[28]   I. Foster, N. T. Karonis, C. Kesselman, and S. Tuecke, "Managing Security in High-Performance Distributed Computing," *Cluster Computing*, vol. 1, pp. 95-107, 1998.
[29]   I. Foster, C. Kesselman, G. Tsudik, and S. Tuecke, "A Security Architecture for Computational Grids," presented at Proc. 5th ACM Conference on Computer and Communications Security Conference, MISSING LOCATION, 1998.
[30]   I. Foster, C. Kesselman, and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," *Intl. J. Supercomputer Applications*, vol. (to appear), 2001 http://www.globus.org/research/papers/anatomy.pdf.
[31]   T. Howes, M. C. Smith, G. S. Good, and T. A. Howes, *Understanding and Deploying Ldap Directory Services*, January ed: MacMillan, 1999, LDAPbook.
[32]   G. v. Laszewski, S. Fitzgerald, I. Foster, C. Kesselman, W. Smith, and S. Tuecke, "A Directory Service for Configuring High-Performance Distributed Computations," in *Proc. 6th IEEE Symp. on High-Performance Distributed Computing*, 1997, pp. 365-375.
[33]   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," presented at Proceedings of the NASA HPCC/CAS Workshop, 2000, URL MISSING.
[34]   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," presented at Proceedings of the NASA HPCC/CAS Workshop, 2000.

[35]   V. Mann and M. Parashar, "Middleware Support for Global Access to Integrated Computational Collaboratories," presented at Proceedings of 10th IEEE International Symposium on High Performance Distributed Computing, (to appear), San Francisco, CA, 2001.

[36]   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 DataGRID-10-TED-0105, 2001.

[37]   S. S. Rosanna Lee, *JNDI API Tutorial and Reference: Building Directory-Enabled Java Applications*: Addison-Wesley, 2000.

[38]   J. Sang, C. Kim, and I. Lopez, " Developing CORBA-Based Distributed Scientific Applications from Legacy Fortran Programs," 2000, http://accl.lerc.nasa.gov/IPG/CORBA/CAS_corba.pdf.

[39]   G. von Laszewski, I. Foster, J. Gawor, and P. Lane, "A Java Commodity Grid Kit," *Concurrency and Computation: Practice and Experience*, vol. 13, pp. 643-662, 2001 http://www.globus.org/cog/documentation/papers/cog-cpe-final.pdf.

[40]   G. von Laszewski, I. Foster, J. Gawor, W. Smith, and S. Tuecke, "CoG Kits: A Bridge between Commodity Distributed Computing and   High-Performance Grids," in *ACM 2000 Java Grande Conference*. San Francisco, CA, 2000, pp. 97--106, http://www.mcs.anl.gov/~laszewsk/papers/cog-final.pdf.

[41]   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*, vol. 72, pp. 2062-2068, 2001.