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

Features of the Java Commodity Grid Kit

Gregor von Laszewski, Jarek Gawor, Peter Lane, Nell Rehn, and Mike Russell
Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL, 60439, U.S.A.

Corresponding Author: Gregor von Laszewski, gregor@mcs.anl.gov, phone: + 630 378 0837 fax: + 630 252 5986
Note: The paper follows the template given in the GCE working group provided for this special issue to address each issue. If a page limit is require the paper can be shortened.

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

SUMMARY

In this paper we report on the features of the Java Commodity Grid Kit. The Java CoG Kit provides middleware for accessing Grid functionality from the Java framework. Java CoG Kit middleware is general enough to design a variety of advanced Grid applications with quite different user requirements. Access to the Grid is established via Globus protocols, allowing the Java CoG Kit to communicate also with the C Globus reference implementation. Thus, the Java CoG Kit provides Grid developers with the ability to utilize the Grid, as well as numerous additional libraries and frameworks developed by the Java community to enable network, Internet, enterprise, and peer-to-peer computing. A variety of projects have successfully used the client libraries of the Java CoG Kit to access Grids driven by the C Globus software. In this paper we also report on the efforts to develop serverside Java CoG Kit components. As part of this research we have implemented a prototype pure Java resource management system that enables one to run Globus jobs on platforms on which a Java virtual machine is supported, including Windows NT machines.

KEY WORDS: Grid Computing, Globus, Peer-to-Peer, Portal, Java

1. INTRODUCTION

Over the past few years, international groups have initiated research in the area of computational Grids to provide scientists with new modalities required by state-of-the-art scientific application domains. High-end applications using such computational Grids include data-, compute-, and network- intensive applications. Examples range from nanomaterials, structural biology, and chemical engineering to high-energy physics and astrophysics. Many of these applications require the coordinated use of real-time large-scale instrument control and experiment handling, distributed data

Received 23 July 2001
Revised 23 July 2001

sharing among hundreds or even thousands of scientists, petabyte distributed storage facilities, parameter studies, and teraflops of compute power.

All of these applications have in common a complex infrastructure that is difficult to manage. Researchers therefore have been developing services, and portals utilizing these services to facilitate these complex environments, and to hide much of the complexity of the underlying infrastructure. The Globus project provides a small set of useful services, including authentication, remote access to resources, and information services to discover and query such remote resource. Unfortunately, these services may not be compatible with the commodity technologies used for application development by the software engineers and scientists.

To overcome these difficulties, the Commodity Grid project is creating what we call *Co*mmodity *G*rid Toolkits (CoG Kits) that define mappings and interfaces between Grid services and particular commodity frameworks. Technologies and frameworks of interest include Java, Python, CORBA, perl, .NET, JXTA.

In this paper we concentrate on the features of the Java CoG Kit. It provides convenient access to Grid functionality through pure Java client-side classes and components. We are also developing pure Java server side components. Although the Java CoG Kit can be classified as middleware for integrating advanced Grid services, it can also be classified, both as a system providing unique advanced services currently not available in Globus and as a framework for designing computing portals.

Grids

The term "Grid" emerged in the past decade and denotes an integrated distributed computing infrastructure for advanced science and engineering applications. The Grid concept is based on coordinated resource sharing and problem solving in dynamic multi-institutional virtual organizations [28]. Besides access to a diverse set of remote resources among different organizations, Grid computing is required to facilitate highly flexible sharing relationships among them, ranging from client-server to peer-to-peer computing. An example of a typical Grid client-server relationship is a supercomputer center to 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 [59][9].

Grids must support different levels of control, ranging from fine-grained access control to delegation, single user to multi-user, and different quality of service mechanisms such as scheduling, coallocation, and accounting. These requirements are not sufficiently addressed by current commodity technologies. 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 some, services and tools that address 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.

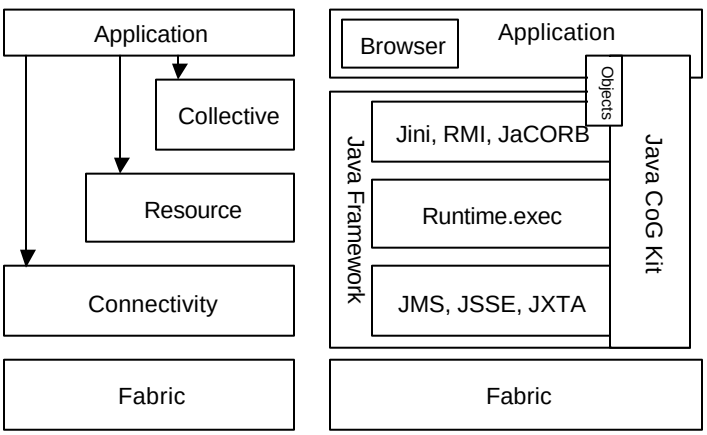

Figure 1: The figure on the left depicts the current view of the Grid architecture. The figure on the right depicts how a subset of Java technologies fits within this architecture. Naturally, this is just a small subset; many more Java technologies exist addressing Grid-related issues, as explained in the text.

Because of the diversity of the Grid, however it is difficult to develop an all-encompassing Grid architecture. Recently, a Grid architecture representation has been proposed [59] that comprises five layers:

- ***fabric layer***, which interfaces to local control, including physical and logical resources such as files or even a distributed file system;
- ***connectivity layer***, which defines core communication and authentication protocols supporting Grid-specific network transactions;
- ***resource layer***, which allows the sharing of a single resource while using a
- ***collective layer*** which allows resources to be viewed as collections; and
- ***application layer,*** which uses the appropriate components of each layer to support the application.

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

Why Java for Grids?

Java has most recently received considerable attention by the Grid community in the area of application integration and portal development. For example, the EU DataGrid effort recently defined Java, in addition to C, as one of their target implementation languages [4]. Several factors make Java a good choice for Grid computing: Java is a modern, object-oriented programming language that makes software engineering of large-scale distributed systems much easier. Thus, it is well suited as a basis for an interoperability framework. Java also has the advantage of platform independence because of its intermediate bytecode representation. This feature allows any pure Java application to run without recompilation on any system that supports a Java 2 Virtual Machine (JVM). Since platform independence is important when delivering applications in heterogeneous environments such as Grids, Java has a big advantage for Grid developers and users. Moreover, the support of Java virtual machines on many client-based systems makes Java an ideal starting point for developing many client side applications. The Java environment includes core libraries that implement common Internet protocols and functionality. Among them are rich and easy-to-use networking libraries that are of particular use in the Grid environment. Java is type safe, has array bounds checking, and sandboxing of running applications. Java also allows for explicit security, providing security APIs that allow for authentication, data integrity, and confidentiality. Furthermore, Java has a number of technologies that are advantageous

for Grid developers. Some of these technologies, which could be layered either above or below Globus, are JAAS [48], JINI [26], JXTA [32], JNDI [51], JSP [35], EJBs [37], and CORBA/IIOP [54] interoperability. For example, the Java Authentication and Authorization Service (JAAS) enables fine-grained and extensible access control, based on who signed a particular code and/or who runs this code. A small subset of Java technologies is mapped in Figure 1 in the corresponding Grid layers. We see that Java provides supporting technologies on many levels within this Grid architecture. Of big advantage are those technologies that are accessed to write user interface applications, object technologies enabling remote object frameworks such as CORBA [57], RMI, and Jini. JIT (just-in-time) technology also has helped dispel the myths about Java's performance being poor when compared with compiled languages such as C or Fortran. With JIT technology, Java bytecode is compiled into native code just before execution, resulting in a significant performance gain coming close to speeds of classical compilers. Furthermore, IBM's research JVM [38] has demonstrated that Java can outperform C and FORTRAN compilers even for numerical calculations. Additional reasons for choosing Java for Grid computing can be found in [47] [31].

Web Technologies for Grids

Web technologies are playing an enormous role in developing future Grid applications and Java is quite well situated as a development framework for Web applications. Accessing technologies such as XML [36], XML schema [18], SOAP [25], and WSDL [22] will become increasingly important, not just for the Web but also for the Grid community. Specifically, by leverage existing technologies as part of ongoing Grid activities as well as simplifying interaction between Grid and Web services. We are currently investigating these and other technologies for Grid computing as part of the commodity Grid projects to create CoGs Kits based on Python, CORBA, Java, and Java Server Pages.

The Java CoG Kit provides a good technology integration framework (see Figure 2) for enabling request driven semantic Grids [53], which combine features of each technology. Java is used to implement many of the Standards proposed by W3C and OMG. Standards promoted through the Global Grid Forum (GGF) are implemented as part of the Java CoG Kit. As a result we will be able to use the Java CoG Kit for interfacing and reusing technologies from the semantic Web and model driven

architectures. Some of the promises of a federation of these technologies as part of a request driven semantic Grid include:

- Provide scalability on various levels including the infrastructure and user base
- Use natural language for interacting with the Grid
- Federate namespaces between ontologies
- Deliver performance
- Provide a request driven behaviour
- React to dynamic behaviour
- Support of commercial platforms

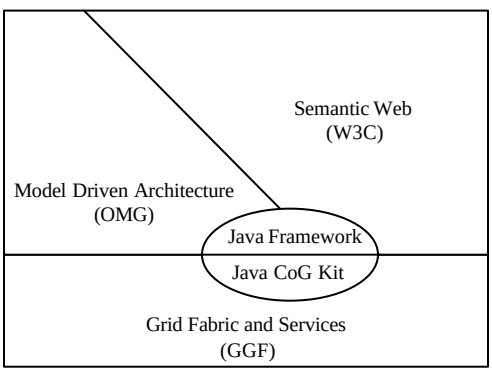

Figure 2: The Java framework is used to build the necessary technology bridge between model driven architectures and the semantic Web. The Java CoG Kit provides the bridge to Grid technologies.

2. OVERVIEW OF THE JAVA COG KIT

Description and Goals

Java CoG Kit [46] [45] [47] is an implementation of Globus protocols and functionality in Java. The components provided are not limited to those of the C Globus implementation [6]. The Java CoG Kit is general enough to be used in the design of a variety of advanced Grid applications with quite different user requirements. The access to the Grid is established via Globus protocols allowing communication with the

C Globus reference implementation. The goal of the Java CoG Kit is to provide Grid developers with the advantage to utilize much of the Globus functionality, as well as, access to the numerous additional libraries and frameworks developed by the Java community allowing network, Internet, enterprise, and peer-to-peer computing. We are currently extending our efforts to also include a pure Java resource management system that enables Grid users to run Globus jobs on platforms on which a Java Virtual Machine is supported. Naturally, this includes machines running Windows NT.

Elementary Grid Services and Features

Grid services that can be accessed through the client-side Java CoG Kit include

- An information service compatible to the Globus MDS [43] implemented with JNDI.
- A security infrastructure compatible to the Globus GSI implemented with the iaik security library [9].
- A data transfer compatible with a subset of the Globus GridFTP and/or GSIFTP is supported.
- Resource management and job submission to the Globus GRAM Gatekeeper [23].
- Quality of service compatible with Globus GARA [30].
- A certificate store based on the myProxy server[49].

The Globus collocation client service DUROC is not supported, because of the planned removal of the Nexus [29] [27] protocol from its dependency list. Thus, MPICH/G2 [39] jobs can be started from a CoG client, but the job must be executed on a Globus C server.

New Services

Besides these elementary Grid services, several other features and services currently not provided by the Globus Toolkit are included explicitly or implicitly within the Java CoG Kit.

The Java Webstart [14] and signed *applet* technologies provide developers with an advanced service to simplify code startup, code distribution, and code update. Java Webstart allows the easy distribution of the code as part of downloadable jar files that

are installed locally on a machine through a browser or an application interface. We demonstrate the use of Webstart within the Java CoG Kit by providing sophisticated Swing applications. One example is the LDAP browser we have developed, which is available via Webstart or can be executed through signed applet technologies at the same Web location [41].

Another well-known feature of Java is the ability to easily define *graphical user interfaces* (GUIs) as part of Java applications and applets. The integration of this feature within the frameworks adhered by vendors is advantageous for developing cross-platform portable JavaBeans. Because of this standardization of the GUI APIs within Java and the existing component model based on the JavaBeans framework, many commercial vendors provide *interface development environments* (IDEs) [13]. Thus, while developing JavaBeans with graphical and non-graphical Grid components, it is already possible to use commercial IDEs as Grid IDEs.

Since many Web browsers support the execution of Java applets and applications (through Webstart with the proper authentication), the placement of Grid services on the client and server side as part of a Web-based strategy is appropriate and desirable. Moreover it can dramatically cut cost in installing and maintaining Grid client software. Similar arguments can be made for other Grid services, as we are currently exploring with a pure Java resource management service. Figure 3 depicts a subset of these technologies that may be reused by the application teams.

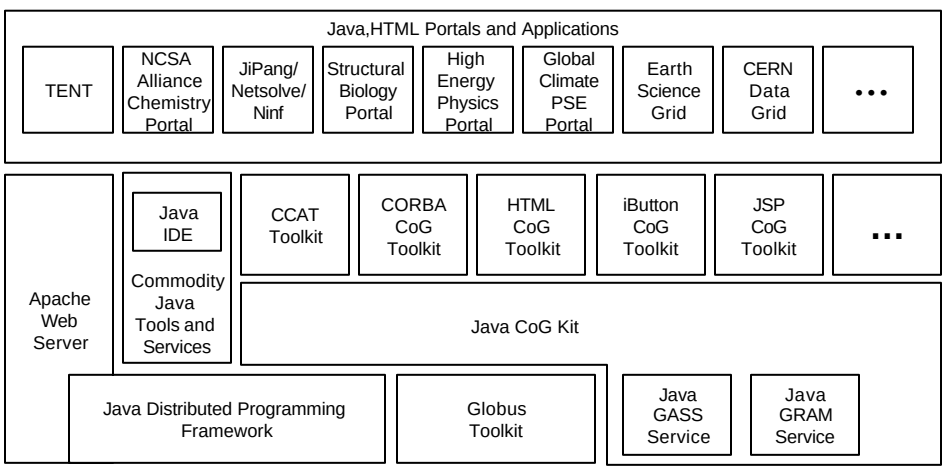

Figure 3: The Java CoG Kit builds a solid foundation for developing Grid applications based on the ability to combine Grid and Web technologies.

Systems, Sites, and Users Served

The user community served by the Java CoG Kit is quite diverse. The Java CoG Kit allows

- *middleware developers* to create new middleware components that depend on the Java CoG Kit such as GPDK [8] and the Rutgers/ANL CORBA CoG Kit [57];
- *portal developers* to create portals that expose transparently the Grid functionality as part of a portal service such as the CoG Box [56]; and
- *application developers* to use of Grid services within the application portal such as the ASC project [3].

A variety of existing projects have successfully used the Java CoG Kit to access Grids driven by the C Globus software. The middleware characteristics make the Java CoG Kit useful as a toolkit to develop Grid client- and server-side applications in pure Java. The Java CoG Kit is particularly well suited for the development of advanced Grid services and Portals. Projects using currently the Java CoG Kit for accessing Grid functionality include the following:

- *CogBox* [56] provides a simple GUI for much of the client-side functionality such as file transfer and job submission.
- *CCAT* [11] provides an implementation of a standard suggested by the Common Component Architecture Forum, defining a minimal set of standard features that a high-performance component framework has to provide, or can expect, in order to be able to use components developed within different frameworks.
- Grid Portal Development Kit (GPDK) [8] provides access to Grid services by using Java Server Pages (JSP) and JavaBeans using Tomcat, a Web application server.
- JiPANG (Jini-based Portal AugmeNting Grids) [55] is a computing portal system that provides uniform access layer to a large variety of Grid services including other PSEs, libraries, and applications.
- The NASA IPG LaunchPad [15] uses the Grid Portal Development Kit based on the Java CoG Kit. The tool consists of easy-to-use windows for users to input job information, such as the amount of memory and number of processors needed.

- The NCSA Science Portal [19, 40] provides a "personal Web server" that the user runs on a workstation. This server has been extended in several ways to allow the user to access Grid resources from a Web browser or from desktop applications.
- The Astrophysics Simulation Code Portal (ASC Portal) [3] is to build a computational collaboratory to bring the numerical treatment of the Einstein theory of general relativity to astrophysics.
- TENT [17] is a distributed simulation and integration system used for example for airplane design in commercial settings.
- ProActive [20] [21] is a Java library for Parallel, Distributed, and Concurrent computing and programming. The library is based on a reduced set of rather simple primitives and supports an active object model. It is based on the RMI Java standard library. The CoG Kit provides access to the Grid.
- DISCOVER [52] is developing a generic framework for interactive steering of scientific applications and collaborative visualization of data sets generated by such simulations. Access to the Grid will be enabled through the CORBA [57] and Java Commodity Grid Kits.
- The Java CORBA CoG Kit [50] [57] provides a simple Grid domain that can be accessed from CORBA clients. The domain is provided as pure Java prototype. Future implementations in C++ are possible.

A regularly updated list of such projects can be found at [42]. We encourage the community to notify us of additional projects using the Java CoG Kit, so we can continue to update the Web page.

3. ARCHITECTURE

The Java CoG Kit integrates Java and Grid components and services within one toolkit, as a *bag* of services and components. In general, each developer chooses the components, services, and classes that ultimately support his development requirements. Java CoG Kit components related to Grid services do not provide a simple one to one mapping between the C Globus and Java CoG Kit API. As a simple example, in the Globus Toolkit on which we are building our prototypes, remote computation management is handled via a procedural API and callbacks; in the Java CoG Kit, the same functionality is provided via a job object and Java events. The services and components within the CoG Kit belong in one of four groups each of which contains a servicerelated implementations application portal and interfaces,

advanced portal and Grid services, mapping and interfaces to basic Grid services, and Grid Services.

The functionality of these Grid components enables mappings and interfaces to both existing and new Grid services. These services build the basis for the development of more advanced services and components. Additionally, we provide a simple set of graphical components that we hope to extend with the input of the user community to develop application portals. Thus, our bag of services can be reused to define architectures based on the application domain. As an example we show in Figure 3 a conceptual diagram for a Grid based computing portal developed with parts of the Java CoG Kit. As the figure indicates services developed by other groups can be integrated into the general portal architecture. Such services include collaborative session management (e.g. for the Access Grid [1]), problem session management, and data pedigree, data management and job management (e.g. by TENT [17]). We note that the Grid services in the lowest level of Figure 4 are currently implemented in C, but in principle there is no restriction that these services could not be provided in pure Java.

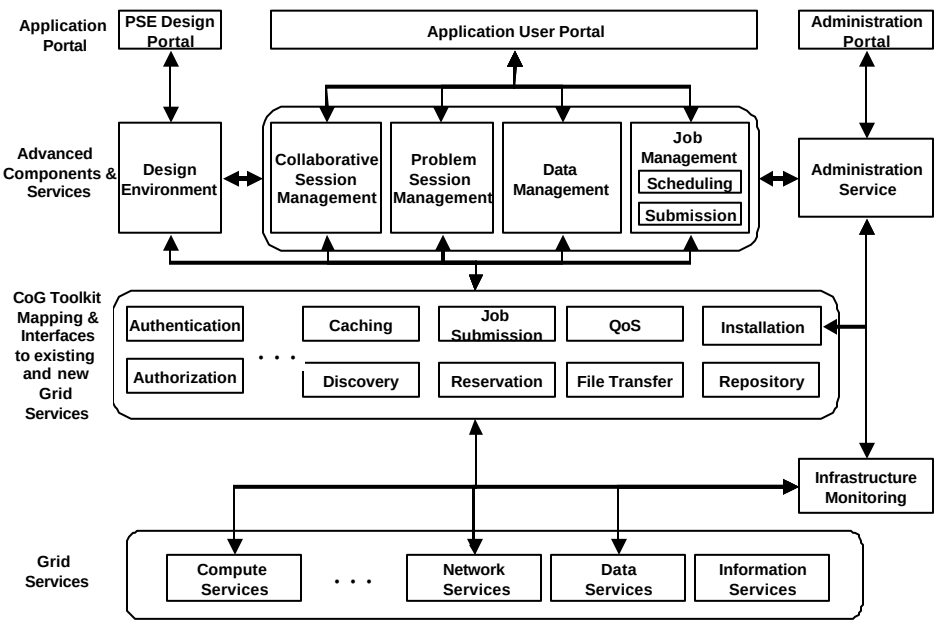

Figure 4: The architecture of an application portal that can be developed with current and future CoG Kit components. Additional components may also be provided by other projects.

Dependencies on Grid Software and Services

Since much of the current Java CoG Kit implements a client-side solution for accessing Grid services of the Globus Toolkit, Globus should be installed on the server to which we want to connect. Nevertheless, for many applications it is possible to install just a subset of Globus services as required by some applications. One can, for example, use the LDAP browser for browsing the MDS without having to install the rest of Globus. In future versions the dependency on the installation of the C Globus version will be relaxed as more and more services are provided as pure Java implementation. We emphasize that the client side of the Java CoG Kit is not dependent on any additional services on the client side. Thus, the download of the jar file is sufficient if a Java virtual machine is present.

Use of Grid software and Services

The Java CoG Kit monitors closely the development within the Globus project to assure that a level of interoperability is maintained. The CoG Kit development team continues to keep track of projects that reuse the Java CoG Kit and documents the requirements of the community, in order to feed this information back to the Globus development team and to develop new features within the Java CoG Kit.

Software Services not Supported by Other Grid Technologies

In addition to the already mentioned services, we provide services that use the Grid Object Specification (GOS) as defined by the Grid Information Services Working group of the Global Grid Forum [5]. A binding from GOS to the RFC2252 [58] also is provided that can be used to create schemas for the Metacomputing Directory Service [43] of the Globus project. Furthermore, we will define simple services using part of the Java CoG Kit to perform the task of a portal to do job submissions and to coordinate tasks via workflows. These examples can be used by the community to create more sophisticated components by the community that we hope to integrate in an open source distribution of the Java CoG Kit.

Optional Components and Services Used by the Java CoG Kit

In addition to the components and services available or accessible through the Java framework, we may use commodity technologies such as XML, Web servers, and Java servlet engines. The use of these components depends on the application domain reusing the Java CoG Kit. An additional useful service is the ability to use smart card or Java iButton technology [10] to perform secure authentication with a possible multiple credential store on a smart card or an iButton.

Impact on the GGF GCE Working Group

The Grid Computing Environments Working Group as part of the Global Grid Forum [33] is exploring the possibility of creating easy to-use and transparent compute environments. Since the goal of the Java CoG Kit is to provide middleware to design such environments it all services and components of the Java CoG Kit clearly are reusable by the GCE Working Group participants.

4. IMPLEMENTATION DETAILS

We are designing the Java CoG Kit based on the usual software engineering practices followed in the Java community.

Commodity Technologies and Software

We are using the Java framework delivered by Sun Microsystems. This includes JNDI, which is included in the J2EE. We use XML parsers, iButtons libraries, and the IAIK Security libraries to implement GSI over SSL. If desired, the creation of customized jar files can be controlled via GNU autoconfigure [7] or ant [2], which is also available for the Windows operating system. We use jUnit for the execution of simple tests for the code. We note that not all technologies need be used to integrate the Java CoG Kit within a users application framework; the user may collect the desired functionality, thus determining the technologies used.

Current Products and Deliverables

The current deliverables of the Java CoG team are as follows:

1. An open source Java code mapping Grid functionality into the Java framework.

2. An extensive set of documentation including a manual, documented examples, and tutorials.
3. A distribution of the Java CoG Kit as source with a low overhead but flexible configuration service to also create custom jar files.
4. A distribution of the Java CoG Kit as jar file with an easy installation service.
5. A gradual inclusion of new components and services as they become available.

Additionally, we have will provide

1. a jar file that includes a GOS conversion tool as part of the GGF requirements [34].
2. a service that performs the GOS conversion through a Web interface.
3. updates to the jar file distribution of the LDAP browser.

The LDAP browser can be freely used for educational purposes under proper acknowledgment. This part of the Java CoG Kit is used over 1,600 times as part of commercial licensing agreements.

Table 1: Classification of the Java CoG Kit classes to Grid services

Grid Service	Class Name
Security	org.globus.security org.globus.myproxy org.globus.iButton
Resource Management / Scheduling	org.globus.rsl org.globus.gram org.globus.server.gram
Information Services	org.globus.mds org.globus.gos org.globus.gosml org.globus.mdsml (deprecated)
Data Transfer	org.globus.io.gass org.globus.io.ftp org.globus.io.urlcopy
Resource Management / Quality of Service	org.globus.gara

Grid Services

Table 1 shows a simple classification between Grid services and the current list of classes that are included in the beta release of the Java CoG Kit. We provide for each class a short technical description of the library, comparing it with the C Globus implementation. For more information we refer to the Web page [41]. Furthermore, we have presented in [45], [46], and [44] selected examples for the usage of these classes.

GSI (org.globus.security)

This library is a partial implementation of GSI. It is fully compatible with Globus GSI and can be used to write GSI-enabled clients and servers. It supports both host and subject authorizations. It does not, however, implement the GAA or offer a GSS API interface. Moreover, it currently does not support certificate revocation lists (CRL) and does not check certificate extensions. One unique feature of the library is that it can manage multiple credentials at the same time in the same process. Any library that uses the Java GSI library can take advantage of this capability.

MyProxy (org.globus.myproxy)

This library provides the MyProxy client API in Java. It is fully compatible with the C implementation of MyProxy. It allows for uploading the Globus credentials to a MyProxy server, retrieving the stored credentials from the server, and destroying them. The C Globus distribution does not include a corresponding library. In many cases using iButton or smart card technologies are preferred and more desirable.

RSL (org.globus.rsl)

This library provides an API for creating, manipulating, and checking the validity of RSL expressions. It also handles an XML-based RSL representation.

GRAM (org.globus.gram)

This library is a full implementation of the GRAM client API. It allows for submitting and canceling of jobs, polling for job status, and sending signals to a job. The library enables a user to 'ping' a gatekeeper to verify whether the user can authenticate to it. In addition, this library allows for registering and unregistering of callback listeners that listen for job status updates. The callbacks are implemented as Java events. Beyond the functionality of Globus, the Java GRAM API allows the specification of the security delegation type to perform, either full or limited.

MDS (org.globus.mds)

This library provides convenient APIs for accessing the MDS service. It allows for querying the MDS and adding, modifying, or deleting MDS entries. The library is based on the JNDI library and enables communication with the MDS over a SASL interface (secure MDS) from both JNDI and the Netscape Directory SDK.

MDSML (org.globus.mdsml)

This library provides an API for converting various schemas to and from the MDSMLv1 format. For example, an MDSML document can be translated into the LDAPv3 schema format or DSML. Work has already been approved for, among other related activities, upgrading this library to conform to the MDSMLv2 specification. The C Globus distribution does not include a corresponding library. This class will be deprecated shortly in favour of the development of gosML together with the GGF. We will keep the code for MDSML in our distribution to provide backwards compatibility but will offer limited support.

GASS (org.globus.io.gass)

This library provides client and server GASS functionality. Java GASS implementation is fully compatible with Globus GASS. It allows, for example, a Java GASS client to connect and transfer a file from a Globus GASS server; or a Globus GASS client to connect and transfer a file from a Java GASS server. The Java GASS client provides the file-access API, while the Java GASS server provides the 'server-ez' API. Java CoG Kit does not support the cache management functionality at this point; nor does it follow the full client and server C API.

GSIFTP (org.globus.io.ftp)

This library provides a client API for accessing and transferring files from GSI-enabled FTP servers. It provides all the common FTP commands as methods, implements more advanced functionality such as recursive file transfers (transferring of entire directories), and supports third-party transfers. It does not, however, support any other advanced GridFTP functionality at this point.

UrlCopy (org.globus.io.urlcopy)

This library provides a simple API for transferring a file from one location to another. The locations are specified as URLs, and any combination of the following protocols is

supported: HTTP, HTTPS, FTP, GSIFTP, and FILE. Also, third party transfers can be initialized between any ftp servers that support that feature.

GARA (org.globus.gara)

This library is an implementation of the GARA reservation API in Java. It allows for creating, cancelling, and modifying various types of reservations, including network, CPU, and SGI graphics pipe reservations. It also allows for polling reservation status, and for registering and unregistering callback listeners just as in the GRAM library. The C Globus distribution does not include a corresponding library.

Table 2: Classification of the Java CoG Kit command line tools to Grid services

Grid Service	Command Line Tool
Security	grid-proxy-init grid-proxy-destroy grid-cert-info grid-change-pass-phrase myproxy (1) visual-grid-proxy-init (2) (3) visual-myproxy-init (1) (2) (3)
Resource Management / Scheduling	Globusrun GramMultiJobRequest (3)
Information Services	grid-info-search mdsml-converter gos-converter
Data Transfer	globus-url-copy globus-gass-server globus-gass-server-shutdown
Resource Management / Quality of Service	

(1) myProxy is a package that is supplied outside of Globus. If possible the use of iButtons or smart cards is preferred.
(2) graphical components
(3) not provided by C Globus

Command-Line Tool Interfaces to Grid Services

The Java Cog Kit contains a set of command-line scripts (Table 2) that provide client-side functionality similar to that in C Globus. The command-line tools include, at present, Bourne shell scripts and Windows batch files, which are thin wrappers around the Java classes that implement the desired functionality. All of these command-line tools mimic the functionality of the appropriate command line tools distributed with the C Globus Toolkit with the exception of g*lobusrun*, which does not support multi-requests (see GramMultiJobRequest through DUROC [24], and *grid-proxy-init*, which does not mask the entry of one's passphrase, since echoing of streams to stdout is not defined as part of the language standard. A GUI version called visual-grid-proxy-init is provided that properly masks an entered passphrase.

Comannd-line services that are not supported by the C version of Globus are the visual components, such as *visual-grid-proxy-init,* providing the same functionality as the grid-proxy-init call, but allows graphical manipulation of proxy settings and providing a masked passphrase entry field. Additionally, we provide means for submitting GlobusMultiJobRequests that do not require the use of DUROC [24]. Thus, it is possible to start a set of related jobs on a set of remote machines as specifiable via the Globus RSL.

Table 3: Classification of the Java CoG Kit classes to Grid services

Grid Service	Computing Portal Component
Security	org.globus.myproxy
Resource Management / Scheduling	org.computingportals.gecco (1) org.computingportals.desktop
Information Services	org.computingportals.mds.MdsTable org.computingportals.mds.Search org.computingportals.mds.SearchTree
Data Transfer	
Resource Management / Quality of Service	org.computingportals.gara.workbench
Installation Service	org.computingportals.common.config

(1) at present, not supported or distributed

GUI Components and Grid Services and Tools

The Java CoG Kit contains a set of graphical components that demonstrate the kits usability as a foundation for graphical Grid applications and that provide a convenient interface to low-level Grid client tools. Table 3 summarizes the available components.

Configuration Wizard

The Configuration Wizard provides the user with a uniform way of configuring the Java CoG Kit. We note that the configuration process is rather trivial, however, and can easily be done manually.

Grid Proxy Init

The Grid Proxy Init component provides a visual interface for creating Globus proxies (see visual-grid-proxy-init).

MyProxy

This component is used to manage proxies on the MyProxy Servers.

GridDesktop

The Grid Desktop is a next-generation portal application that demonstrates the benefits of Java, the Grid, and Web-based portals. Currently, this component is implemented as an application. This component can also be used to demonstrate the integration of drag-n-drop functionality into Grid applications.

MDS Components

These components are simple examples and can be used to develop more sophisticated components.

GARA Workbench

This application provides a simple user interface for managing network reservations. It has been shown as a demo at SC2000 [12].

Other Features

Other implementation related features of the Java CoG Kit includes firewall support with the ability to set the port ranges for machines behind firewalls or NAT servers. To increase the performance of the authentication process on Linux machines, one can use

the /dev/urandom device to create the necessary random numbers for faster seed generation as needed by the GSI library implementation.

5. FUTURE WORK

Java Gram Service

One of the additional features we planed for the Java CoG Kit is a pure Java service that provides the ability to submit RSL strings to this service and execute them in a fashion similar to that of the C Globus GRAM service. The implementation of such a service is based on a generalized multithreaded server. This server, originally used to implement the GASS server provided by the Java CoG Kit, can be reused to implement the GRAM server, because the object oriented design means that the protocol used on the incoming socket connection can be changed easily. Internally, we distinguish the following components (Figure 5). A multithreaded *gatekeeper* accepts incoming connections from the client software. The RSL string part of the message gets interpreted, and a job manager gets started that works as proxy between the client and the environment executing the job. If the job specifies an executable it gets executed through the Java Runtime.exec() call. The I/O can be appropriately redirected to and from the client dependent on the specification within the RSL string. The gatekeeper and the job manager log their state changes into a log file in order to provide accounting information. Besides the execution of executables for the specified operating system, we can also execute signed and unsigned Java programs submitted as jar files as part of our GRAM server. The GRAM server can be configured in one of two ways: the program associated job manager gets executed in the same virtual machine as the gatekeeper, or each job manager gets executed in an own virtual machine (Figure 6). We are currently experimenting with these setups to provide a performance vs. security level ability that will be beneficial once we have integrated the Globus map file mechanism. In contrast to the current C Globus GRAM service, we are also investigating the support of the gsiftp protocol as part of the service. This will enable us to use gsiftp as part of the redirection capabilities for I/O; the GRAM service uses GSI for authentication.

The reasons for the developing such a GRAM service are manifold. First, it is possible to develop a service similar to seti@home [16], but with the advantage that applications can be augmented with the usual Java properties to enable secure access within the Java

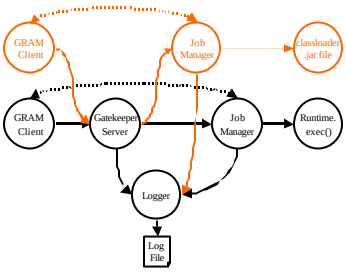

Figure 5: Overview of the components constituting the pure Java GRAM service.

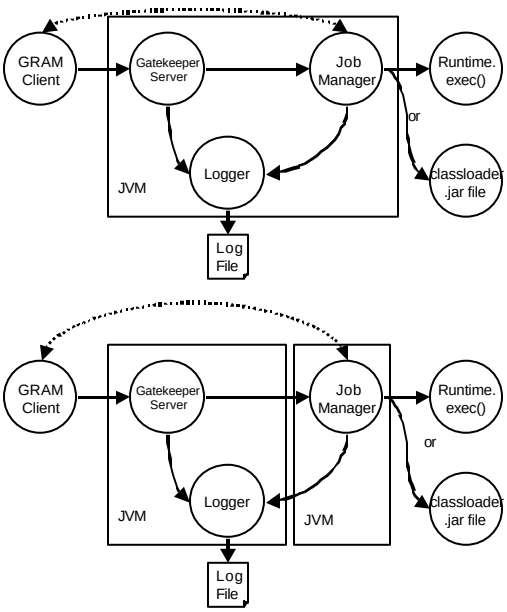

Figure 6: The GRAM service can be started either within the same JVM or within a different one to increase fault tolerance and security.

sandbox model. Thus, Java provides implicitly an additional security service for building aggregations of compute services. This service together with the MDS can be used to build a virtual community. The model to describe this community is built on two roles: compute providers and compute consumers. The providers register their resources with the virtual organization, and the consumers register their tasks through the submission of jar files and meta-information about the job. A broker (based, for example, on Condor matchmaking) could provide for an easy assignment of jobs to the resources. Although many other use models are possible, we have chosen this model because of its well-known characteristics within the Grid community.

6. CONCLUSION

Commodity distributed-computing technologies enable the rapid construction of sophisticated client-server applications. Grid technologies provide advanced network services for large-scale, wide area, multi-institutional environments and for applications that require the coordinated use of multiple resources. In the Commodity Grid project, we bridge these two worlds so as to enable advanced applications that can benefit from both Grid services and sophisticated commodity development environments.

The Java Commodity Grid Project is creating such a bridge for the Java framework. We provide an elementary set of classes that allow the Java programmer to access basic Grid services, as well as enhanced services suitable for the definition of desktop problem solving environments.

The Java CoG Kit is available as a beta version from the CoG Kit Web pages [41]. It has already received considerable testing by external projects. The CoG Kit project will provide additional services for development of Grid applications. These additional Java CoG Kit components will be released gradually while adhering to quality control standards. Some components described in this paper are not officially released yet. For up-to-date release notes, readers should refer to the Web page at http://www.globus.org/cog. New releases are announced to the mailing list at cog-news@globus.org.

Our future work will involve the integration of more advanced services into the Java CoG Kit and the creation of other CoG Kits, with CORBA and Python being short-term priorities. We hope to gain a better understanding of where changes to commodity or Grid technologies can facilitate interoperability and of how commodity technologies can be exploited in Grid environments.

ACKNOWLEDGEMENTS

This work was supported by the Mathematical, Information, and Computational Science Division subprogram of the Office of Advanced Scientific Computing Research, U.S. Department of Energy, under Contract W-31-109-Eng-38. DARPA, DOE, and NSF support Globus research and development. We thank Ian Foster, Geoffrey C. Fox, Dennis Gannon, and Jay Alameda, for the valuable discussions during the course of the ongoing CoG Kit development. This work would not have been possible without the help of the Globus team.

REFERENCES

[1] "Access Grid Web Page," 2001, http://www-fp.mcs.anl.gov/fl/accessgrid/.
[2] "ant: A Jakarta Project," 2001, http://jakarta.apache.org/ant/index.html.
[3] "The Astrophysics Simulation Collaboratory: A Laboratory For Large Scale Simulations of Relativistic Astrophysics," 2001, http://www.ascportal.org/.
[4] "Data Grid and Data Managament Project," 2001, http://www.cern.ch/grid.
[5] "Global Grid Forum Web Page," http://www.gridforum.org.
[6] "The Globus project WWW page," 2001.
[7] "GNU Autoconf," 2001, http://www.gnu.org/software/autoconf/autoconf.html.
[8] "The Grid Portal Development Kit," 2000, http://dast.nlanr.net/Projects/GridPortal/.
[9] "IAIK Java Crytology," 2001, http://jcewww.iaik.at/.
[10] "iButton Web Page," 2001, http://www.ibutton.com/.
[11] "Indiana CCAT Home Page," 2001, http://www.extreme.indiana.edu/ccat/.
[12] "Internet Audio Demonstration Wins SC2000 Award," NCSA Access News Brief, 2000, http://www.ncsa.uiuc.edu/News/Access/Briefs/00Briefs/001212.SC2000.html.

[13] "Java Forte Integrated Developement Environment," 2001, http://www.sun.com.
[14] "Java Web Start Web Page," Version 1.0.1 ed, 2001, http://java.sun.com/products/javawebstart/.
[15] "Launching into Grid Space with the NASA IPG Launchpad," 2001, http://www.nas.nasa.gov/Main/Features/2001/Winter/launchpad.html.
[16] "Seti at Home Web Page," 2001, http://setiathome.ssl.berkeley.edu/.
[17] "TENT Home Page," German Air and Space Agency (DLR), 2001, http://www.sistec.dlr.de/tent/.
[18] "XML SChema, Primer 0 - 3," 2001, http://www.w3.org/XML/Schema.
[19] J. Alameda, "Chemical ENgeneering Portal," 2001, http://www.ncsa.uiuc.edu/Science/ChemEng/.
[20] D. Caromel, "ProACtive Java Library for Parallel, Distributed and Concurrent Programmming," 2001, http://www-sop.inria.fr/oasis/ProActive/.
[21] D. Caromel, W. Klauser, and J. Vayssiere, "Towards Seamless Computing and Metacomputing in Java," *Concurrency Practice and Experience*, vol. 10, pp. 1043--1061, 1998 http://www-sop.inria.fr/oasis/ProActive/.
[22] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana, "Web Services Description Language (WSDL) 1.1," W3C Note, 2001, http://www.w3.org/TR/wsdl.
[23] K. Czajkowski, I. Foster, N. Karonis, C. Kesselman, S. Martin, W. Smith, and S. Tuecke, "A Resource Management Architecture for Metacomputing Systems," presented at Proceedings of IPPS/SPDP '98 Workshop on Job Scheduling Strategies for Parallel Processing,, LOCATION MISSING, 1998.
[24] 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.
[25] D. E. Don Box, Gopal Kakivaya, Andrew Layman, and H. F. N. Noah Mendelsohn, Satish Thatte, Dave Winer, "Simple Object Access Protocol (SOAP) 1.1," 2000, http://www.w3.org/TR/SOAP.
[26] W. K. Edwards, *Core Jini*, 2nd edition ed: Prentice Hall Computer Books, 2000.
[27] I. Foster, J. Geisler, W. Gropp, N. Karonis, E. Lusk, G. Thiruvathukal, and S. Tuecke, "A Wide-Area Implementation of the Message Passing Interface," *Parallel Computing*, vol. 24, pp. 1735--1749, 1998.
[28] 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.

[29] I. Foster, C. Kesselman, and S. Tuecke, "The Nexus Approach to Integrating Multithreading and Communication," *Journal of Parallel and Distributed Computing*, vol. 37, pp. 70--82, 1996.
[30] I. Foster, A. Roy, and V. Sander, "A Quality of Service Architecture that Combines Resource Reservation and Application Adaptation," presented at Proc. 8th International Workshop on Quality of Service, 2000.
[31] G. C. Fox and W. Furmanski, "High Performance Commodity Computing," in *The Grid: Bluepirnt for a new computing infrastructure*, I. Foster and C. Kesselman, Eds.: Morgam Kaufman, 1999.
[32] L. Gong, "Project JXTA: A Technology overview," 2001, http://www.jtxa.org.
[33] M. H. Gregor von Laszewski, Steve Fitzgerald, Al Gilman, "GGF GIS Working Group Charter," GWD-GIS-000-3 ed: Global Grid Forum Information Services Working Group, 2001, http://www-unix.mcs.anl.gov/gridforum/gis/.
[34] S. F. Gregor von Laszewski, Pete Vanderbilt, Peter Lane, Brett Didier, "GOSv3: A Data Definition Language for Grid Information Services," Argonne National Laboratory and Pacific Northwest Laboratory, Grid Forum Working Group Document GWD-GIS-011-11, June 2000 2001, http://www-unix.mcs.anl.gov/gridforum/gis/reports/gos-v3/gis-wg-021-002.html.
[35] M. Hall, *Core Servlets and JavaServer Pages (JSP),* 1 edition ed: Prentice Hall PTR/Sun Microsystems Press, 2000.
[36] S. Holzner, *Inside XML*, 1 edition ed: New Riders Publishing, 2000.
[37] D. M. John Crupi, Deepak Alur, *Core J2EE Patterns: Best Practices and Design Strategies*, 1st edition ed: Prentice Hall PTR/Sun Microsystems Press, 2001.
[38] S. M. José Moreira, Manish Gupta, "A Comparison of Three Approaches to Language, Compiler, and Library Support for Multidimensional Arrays in Java," presented at Joint ACM Java Grande - ISCOPE 2001 Conference, Stanford University, 2001.
[39] N. Karonis, "MPICH-G2 Web Page," 2001, http://www.hpclab.niu.edu/mpi/.
[40] S. Krishnan, R. Bramley, D. Gannon, M. Govindaraju, R. Indurkar, A. Slominski, B. Temko, R. Alkire, T. Drews, E. Webb, and J. Alameda, "The XCAT Science Portal," presented at Accepted for Proceedings of SC2001, 2001.
[41] G. v. Laszewski, "The CoG Kit Web Pages," 2001, http://www.globus.org/cog.
[42] G. v. Laszewski, "Projects using the Java CoG Kit," 2001, http://www.globus.org/cog/java/projects.html.

[43] 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.

[44] G. v. Laszewski and I. Foster, "Grid Infrastructure to Support Science Portals for Large Scale Instruments," in *Proc. of the Workshop Distributed Computing on the Web (DCW)*: University of Rostock, Germany, 1999, http://www.mcs.anl.gov/~laszewsk/papers/rostock.pdf.

[45] G. v. 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.

[46] G. v. 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.

[47] G. v. Laszewski, V. Getov, M. Philippsen, and I. Foster, "Multi-Paradigm Communications in Java for Grid Computing," *Communications of ACM*, 2001 http://www.globus.org/cog/documentataion/papers/.

[48] S. Microsystems, "JavaTM Authentication and Authorization Service (JAAS)," vol. 2001, http://java.sun.com/products/jaas/.

[49] J. Novotny, S. Tuecke, and V. Welch, "An Online Credential Repository for the Grid: MyProxy," presented at to be published in HPDC, 2001.

[50] M. Parashar, S. Verma, and G. v. Laszewski, "CORBA CoG Architecture & Implementation," 2001, http://www.caip.rutgers.edu/TASSL/CorbaCoG/CORBACog.htm.

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

[52] R. M. a. M. P. Samian Kaur, "An Environment for Web based Interaction and Steering of Scientific Applications," presented at ACM Java Grande Conference, 2000.

[53] E. V. Schweber, "Summary of the Software Services Grid Workshop."

[54] J. Siegel, *CORBA 3 Fundamentals and Programming*, 2nd Edition ed: John Wiley & SonS, 2000.

[55] T. Suzumura, S. Matsuoka, and H.Nakada, "A Jini-based Computing Portal System," 2001.

[56] B. Temko, "The CoGBox Home Page," 2001, http://www.extreme.indiana.edu/~btemko/cogbox/.

[57] S. Verma, J. Gawor, G. v. Laszewski, and M. Parashar, "A CORBA Commodity Grid Kit," presented at 2nd International Workshop on Grid Computing in conjunction with Supercomputing 2001 (SC2001), Denver, Colorado, 2001.

[58] M. Wahl, A. Coulbeck, T. Howes, and S. Kille, "Lightweight Directory Access Protocol (v3): Attribute Syntax Definitions," 1997, http://www.faqs.org/rfcs/rfc2252.html.

[59] 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.

