CONCURRENCY—PRACTICE AND EXPERIENCE
Concurrency: Pract. Exper. 2000; **00**:1–7

Commodity Grid Kits - Middleware for Building Grid Computing Environments

Gregor von Laszewski[1*†], Jarek Gawor[1], Sriram Krishnan[3,1], and Keith Jackson[2]

[1]*Argonne National Laboratory, 9700 S. Cass Ave., Argonne, IL 60439, U.S.A.*
[2]*Lawrence Berkeley National Laboratory, 1 Cyclotron Rd., Berkeley, CA 94720, U.S.A.*
[3]*Indiana University, 150 S. Woodlawn Ave., Bloomington, IN 47405, U.S.A.*

SUMMARY

Recent Grid projects, such as the Globus Project, provide a set of useful services such as authentication and remote access to resources, and information services to discover and query such remote resources. Unfortunately, these services may not be compatible with the commodity technologies used for application development by the software engineers and scientists. Instead, users may prefer accessing the Grid from a higher level of abstraction than what such toolkits provide. To bridge this gap, Commodity Grid (CoG) Kits provide the middleware for accessing the functionality of the Grid from a variety of commodity technologies, frameworks, and languages. It is important to recognize that these Commodity Grid Kits not only provide an interface to existing Grid technologies, but also bring Grid programming to a new level by leveraging the methodologies of the chosen commodity technology, thus helping the development of the next generation of Grid services. Based on these Commodity Grid Toolkits, a variety of higher level Grid services are far easier to design, maintain, and deploy. Several projects have successfully demonstrated the use of Commodity Grid Kits for the design of advanced Grid Services and Grid Computing Environments.

KEY WORDS: Commodity Grid Kits; Grid; Globus; Problem Solving Environments; Grid Computing Environments

1. INTRODUCTION

Over the past few years, various international groups have initiated research in the area of parallel and distributed computing in order to provide scientists with new programming

*Correspondence to: Argonne National Laboratory, 9700 S. Cass Ave., Argonne, IL 60439, U.S.A.
†E-mail: gregor@mcs.anl.gov

Received 7 July 2002
Revised 7 July 2002

methodologies that are required by state-of-the-art scientific application domains. These methodologies target collaborative, multi-disciplinary, interactive, and large-scale applications that access a variety of high-end resources shared with others. This research has resulted in the creation of computational Grids.

The term "Grid" has been popularized during the past decade and denotes an integrated distributed computing infrastructure for advanced science and engineering applications. The concept of the Grid is based on coordinated resource sharing and problem solving in dynamic multi-institutional virtual organizations [31]. In addition to providing access to a diverse set of remote resources located at different organizations, Grid computing is required to accommodate numerous computing paradigms, ranging from client-server to peer-to-peer computing. High-end applications using such computational Grids include data-, compute-, and network-intensive applications. Application examples range from nanomaterials [33], structural biology [63], and chemical engineering [62], to high-energy physics and astrophysics [48]. Many of these applications require the coordinated use of real-time large-scale instrument and experiment handling, distributed data sharing among hundreds or even thousands of scientists [23], petabyte distributed storage-facilities, and teraflops of compute power. Common to all these applications is a complex infrastructure that is difficult to manage [61]. Researchers therefore have been developing basic and advanced services, and portals for these services, to facilitate the realization of such complex environments and to hide the complexity of the underlying infrastructure. The Globus Project [13] provides a set of basic Grid services, including authentication and remote access to resources, and information services to discover and query such remote resource. However, these services may not be available to the end user at a level of abstraction provided by the commodity technologies that they use for their software development.

To overcome these difficulties, the Commodity Grid project is creating as a community effort what we call Commodity Grid Toolkits (CoG Kits) that define mappings and interfaces between Grid services and particular commodity frameworks. Technologies and frameworks of interest currently include Java [59, 46], Python [41], CORBA [56], Perl [55], and Web Services.

In the following sections we elaborate on our motivation for the design of Commodity Grid Kits. First, we define what we understand by terms such as Grid Computing Environments (GCEs) and Portals. We then illustrate the creation of a GCE with the help of commodity technologies provided through the Java framework. Next, we outline differences from other CoG Kits and provide an overview of ongoing research in the Java CoG Kit Project, which is part of the Globus Project.

2. GRID COMPUTING ENVIRONMENTS AND PORTALS

Grid Computing Environments [2] are aimed at providing scientists and other Grid users with an environment that accesses the Grid by using a coherent and interoperable set of frameworks that include Portals, Problem Solving Environments, and Grid and Commodity Services. This goal is achieved by developing Grid and commodity standards, protocols, APIs, SDKs, and methodologies, while reusing existing ones.

We define the term "Grid Computing Environment" as follows.

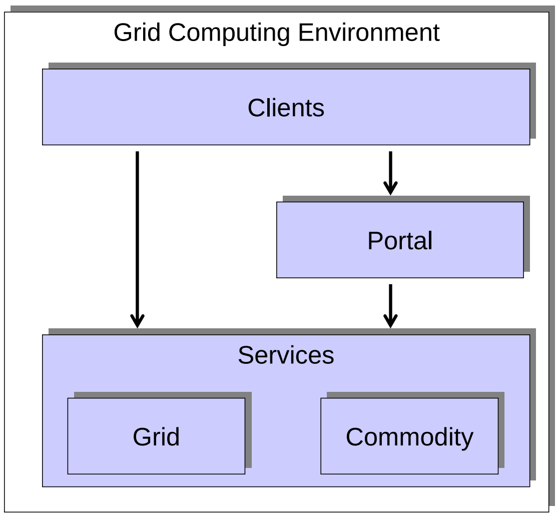


Figure 1. A Grid Computing Environment hides many of the complex interactions between the accessible services.

Definition: Grid Computing Environment

An integrated set of tools that extend the user's computing environment in order to provide access to Grid Services.

Grid Computing Environments include portals, shells, and collaborative and immersive environments running on the user's desktop on common operating systems such as Windows and Linux or on specialized devices ranging from Personal Digital Assistants (PDAs) to virtual reality environments such as stereographic devices or even CAVEs.

The architecture of a GCE can be represented as a multi-tier model. The components of this architecture are shown in Figure 1. Clients access the services through a portal or communicate with them directly. The user is oblivious of the fact that a service may engage other services on his or her behalf.

The term "Portal" is not defined uniformly within the computer science community. Sometimes it represents integrated desktops, electronic market places, or information hubs [34, 51, 35]. We use the term here in the more general sense of a community access point to information and services. Hence, we define the term as follows.

Definition: Portal

A community service with a single point of entry to an integrated system providing access to information, data, applications, and services.

In general, a portal is most useful when designed for a particular community in mind. Today, most *Web Portals* build on the current generation of Web-based commodity technologies, based on the HTTP protocol for accessing the information through a browser.

Definition: Web Portal

A portal providing users ubiquitous access, with the help of Web-based commodity technologies, to information, data, applications, and services.

A *Grid portal* is a specialized portal useful for users of computational Grids. A Grid portal provides information about the status of the Grid resources and services. Commonly this information includes the status of batch queuing systems, load, and network performance between the resources. Furthermore, the Grid portal may provide a targeted access point to useful high-end services, such as a compute and data intensive parameter study for climate change. Grid portals provide communities another advantage: they hide much of the complex logic to drive Grid-related services with simple interaction through the portal interface. Furthermore, they reduce the effort needed to deploy software for accessing resources on computational Grids.

Definition: Grid portal

A specialized portal providing an entry point to the Grid to access applications, services, information, and data available within a Grid.

In contrast to Web portals, Grid portals may not be restricted to simple browser technologies but may use specialized plug-ins or executables to handle the data visualization requirements of, for example, macromolecular displays or three-dimensional high-resolution weather data displays. These custom-designed visual components are frequently installed outside a browser, similar to the installation of MP3 players, PDF browsers, and videoconferencing tools.

Figure 2 presents a more elaborate architecture [60, 61] for representing a GCE that integrates many necessary Grid Services and can be viewed as a basis for many Grid portal activities. We emphasize that special attention must be placed on deployment and administrative services, which are almost always ignored in common portal activities [57]. As shown in the Figure 2, users are interested in services that deal with advanced job management to interface with existing batch queuing systems, to execute jobs in a fault-tolerant and reliable way, and to initiate workflows. Another useful service is reliable data management that transfers files between machines even if a user may not be logged in. Problem session management allows the users to initiate services, checkpoint them, and check on their status at a later time. All of these services are examples of the many possible services in a GCE and are based on the most elementary Grid services. The availability of commodity solutions for installation and rapid prototyping is of utmost importance for acceptance within the demanding user communities.

A Grid portal may deal with different user communities, such as developers, application scientists, administrators, and users. In each case, the portal must support a personal view that remembers the preferred interaction with the portal at the time of entry. To meet the needs of this diverse community, sophisticated Grid portals (currently under development) are providing commodity collaborative tools such as newsreaders, e-mail, chat, video conferencing, and event scheduling. Additionally, some Grid portal developers are exploiting commodity technologies such as JavaBeans and JSP, which are already popular in Web portal environments.

Researchers interested in Grid Computing Environments and Portals can participate in the GCE working group [2] which is part of the Global Grid Forum [1]. The origins of this working

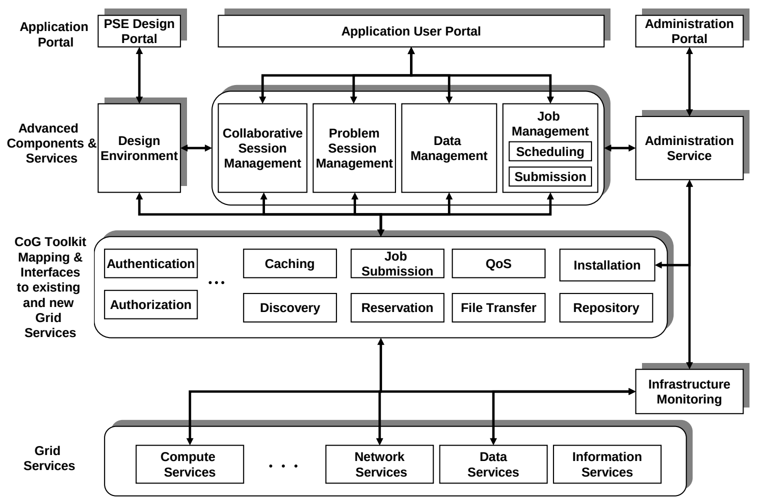


Figure 2. An example of a Grid Computing Environment that integrates basic and advanced Grid and commodity services.

group can be traced back to the Desktop Access to Remote Resources organization that was later renamed to ComputingPortals.org and are spin-offs from the Java Grande Forum efforts [5].

3. COMMODITY TECHNOLOGIES

GCEs are usually developed by reusing a number of commodity technologies that are an integral part of the target environment. For example, a GCE implementing a Web Portal may require the use of protocols such as HTTPS and TCP/IP. It may make use of APIs such as CGI, SDKs such as JDK1.4, and commercial products such as Integrated Development Environments (IDEs) to simplify the development of such an environment. The Grid community has so far focused mostly on the development of protocols and development kits with the goal of defining a standard. This effort has made progress with the introduction of the Global Grid Forum and pioneering projects such as the Globus Project. So far the activities have mostly concentrated on the definition of middleware that is intended to be reused in the design of Grid applications. We believe that it is important to learn from these early experiences and to derive a middleware toolkit for the development of Grid Computing Environments. This is where Commodity Grid Kits come into the picture.

Commodity Grid Kits play the important role of enabling access to the Grid functionality from within the commodity technology chosen to build a Grid Computing Environment. Because of the use of different commodity technologies as part of different application

Table I. A subset of commodity technologies used to develop Grid Computing Environments

	Languages	APIs	SDKs	Protocols	Hosting Environments	Methodologies
Web Portals	Java, Perl, Python	CGI	JDK1.4	HTTPS, TCP/IP, SOAP	JVM, Linux, Windows	OO and Procedural
Desktops	C, C++, VisualBasic, C#	KParts,GTK	KDE, GNOME .NET	CORBA DCOM	Linux, Windows	OO and Procedural
Immersive Environments	C++	CaveLib	Viz5D	TCP/IP	Linux	OO

requirements, a variety of CoG Kits must be supported. In Table I we list a subset of commodity technologies that we have found useful to developing GCEs.

The availability of such CoG Kits is extremely helpful for the Grid application developers as they do not have to worry about the tedious details of interfacing the complex Grid services into the desired commodity technology. As good examples, we present the Java and the Python CoG Kits for the Globus Toolkit, known as Java CoG and pyGlobus, respectively. Both have been used in several GCE developments. However, it is important to recognize the different approaches the Java and the Python CoG Kit pursue.

While the Python CoG Kit interfaces with the Globus Toolkit on an API-based level, the Java CoG Kit interfaces with Globus services on a protocol level. The Python CoG Kit assumes the availability of precompiled Globus Toolkit libraries on the current hosting system, while the Java CoG Kit is implemented in pure Java and does not rely on the C-based Globus Toolkit. Both approaches provide a legitimate approach to achieve Globus Toolkit compliance. Each approach has advantages and disadvantages that are independent from the language chosen. Since the Python interface is generated by using SWIG [24], it is far easier and faster to provide adaptations to a possibly changing toolkit such as the Globus Toolkit. Nevertheless, the price is that the Globus Toolkit libraries must be tightly integrated in the hosting environment in which the Python interpreter is executed. The first version of the Java CoG Kit was based on JNI wrappers for the Globus Toolkit APIs. This approach, however, severely restricted the usage of the Java CoG Kit for developing pure Java clients and portals that are to be executed as part of browser applets. Hence, we implemented the protocols and some major functionality in pure Java in order to provide compliance with the Globus Toolkit. The availability of the functionality of the Globus Toolkit in another language has proved valuable in providing portability and assurance of code quality through protocol compliance.

Both the Python and Java CoG Kits provide additional value to Grids over and above a simple implementation of the Globus Toolkit APIs. The use of the commodity technologies such as object orientation, stream management, sophisticated exception, and event handling enhances the ability to provide the next generation of Grid services. Moreover, in many cases we find it inappropriate to develop such advanced services from scratch if other commodity technologies can be effectively used. A good example is the abstraction found in Java that hides access to databases or directories in general class libraries such as JDBC and JNDI; the absence of such abstractions in other languages might make it more complicated to implement the requisite functionality in such languages.

The availability of a variety of CoG Kits targeting different commodity technologies provides a great deal of flexibility in developing complicated services. We now focus on the Java CoG Kit as an example Commodity Grid Kit, and illustrate how it can be used to effectively build components that can be reused in the implementation of a GCE.

4. OVERVIEW OF THE JAVA COG KIT

Several factors make Java a good choice for Grid Computing Environments. 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 and for exposing the Grid functionality at a higher level of abstraction than is possible with the C Globus Toolkit. Numerous factors such as platform independence, a rich set of class libraries, and related frameworks make Grid programming easier. Such libraries and frameworks include JAAS [52], JINI [29], JXTA [39], JNDI [47], JSP [43], EJBs [44], and CORBA/IIOP [50]. We have depicted in Figure 3 a small subset of the Java technology that can be used to support various levels of the Grid architecture [31]. The Java CoG Kit builds a bridge between existing Grid technologies and the Java framework while enabling each to use the other's services to develop Grid services based on Java technology and to expose higher-level frameworks to the Grid community while providing interoperability [59]. The Java CoG Kit provides convenient access to the functionality of the Grid through client-side and a limited set of server-side classes and components.

Furthermore, Java is well suited as a development framework for Web applications. Accessing technologies such as XML [49], XML schema [15], SOAP [25], and WSDL [26] will become increasingly important for the Grid community. We are currently investigating these and other technologies for Grid computing as part of the Commodity Grid projects to prototype a new generation of Grid services.

Because of these advantages, Java has 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. Additional motivation for choosing Java for Grid computing can be found in [38].

The Java CoG Kit is general enough to be used in the design of a variety of advanced Grid applications with different user requirements. 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 or her development requirements. The goal of the Java CoG Kit is to enable Grid developers to use much of the Globus Toolkit functionality and to have access to the numerous additional libraries and frameworks developed by the Java community, allowing network, internet, enterprise, and peer-to-peer computing. Since the Java CoG Kit strives to be only protocol compliant, it does not provide a simple one-to-one mapping between the C Globus Toolkit and Java CoG Kit API. Instead, it uses the more advanced features of Java,

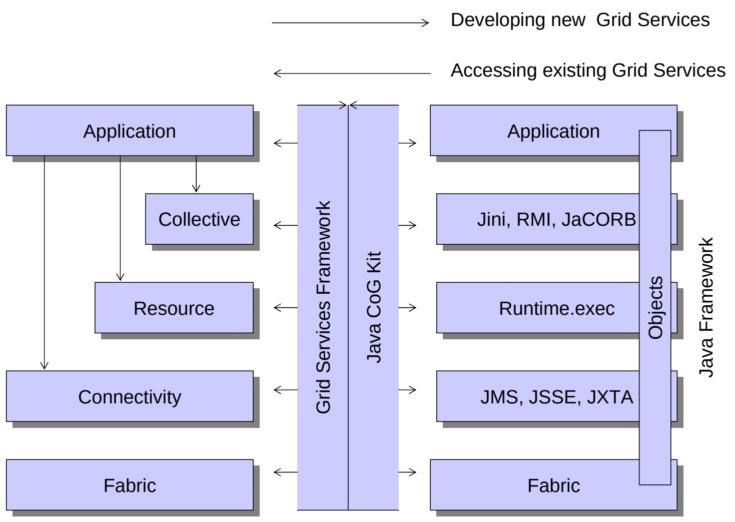


Figure 3. The Java CoG Kit allows users to access Grid Services from the Java framework and enables application and Grid developers to use a higher level of abstraction for developing new Grid services and GCEs

such as the sophisticated Java events and exception handling, rather than using the archaic C-based functions. It provides client side access to the following Grid services :

- An information service compatible with the Globus Toolkit Metacomputing Directory Service (MDS) [58] implemented using JNDI.
- A security infrastructure compatible with the Globus Toolkit Grid Security Infrastructure (GSI) implemented with the IAIK security library [4].
- A data transfer compatible with a subset of the Globus Toolkit GridFTP [19] and/or GSIFTP [20].
- Resource management and job submission to the Globus Resource Access Manager (GRAM) [27].
- A certificate store based on the MyProxy server [45].

Additionally, the Java CoG Kit contains a set of command-line scripts that provide convenient access to Globus Toolkit-enabled production Grids from the client. This set includes support for MS Windows batch files, which are not supported by the C Globus Toolkit. Furthermore, we provide an enhanced version of "globusrun" which allows the submission of multiple GRAM jobs. Other useful services include the ability to access Java smart card or iButton technology [8] to perform secure authentication with a possible multiple credential store on a smart card or an iButton. Besides these elementary Grid services and tools, several other features and services currently not provided by the C Globus Toolkit are included explicitly or implicitly within the Java CoG Kit.

The Java Webstart [10] 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 have demonstrated the use of Webstart within the Java CoG Kit by installing sophisticated Graphical User Interface (GUI) applications on client machines. Component frameworks, such as JavaBeans, and the availability of commercial integrated development environments (IDEs) enable the Grid developer to use IDEs as part of rapid Grid prototyping while enabling code reuse in the attempt to reduce development costs.

Thus, our goal of developing collaborative scientific problem solving environments and portals, based on the combined strength of the Java and the Grid technologies, is well substantiated by the Java CoG Kit. In the past, we had proposed portal architectures similar to the one depicted in Figure 2, in which the Java CoG Kit is used as an elementary middleware to integrate Grid services within portals and applications. We expect that advanced services will be integrated in future releases within the Java CoG Kit or as extension packages. Additionally, it is possible to implement several core Grid services, currently provided as part of the C Globus Toolkit, in pure Java while exposing the service through the Web Services Framework proposed recently by W3C. This possibility has been demonstrated for file transfer and for job execution. The availability of these services and protocol handlers in pure Java will make future portal development and the integration with existing production Grid far easier. We have provided example programs using advanced GUI components in Java as part of the Java CoG Kit. These examples include a setup component for the Java CoG Kit, a form-based job submission component, a drag-and-drop-based submission component similar to a Windows desktop, an information service browser, and search queries. We hope that the community will contribute more components so that the usefulness of the Java CoG Kit will increase.

5. CURRENT WORK

Our current work is focused on the creation of an extended execution service and the integration of Web services in our CoG Kit efforts. Although these are currently prototyped in Java, it is easily possible to provide implementations in other languages like C and C++.

5.1. InfoGram

An important result from this prototyping has been the development of the "InfoGram" service, which integrates a job submission service and an information service into a single service while reducing the development complexity. This InfoGram service has been described in more detail in [64] outlining extensions to the Globus Resources Specification Language (RSL) [22] and the integration of checkpointing. Currently, we are also exploring the use of the InfoGram Service as part of "Sporadic Grids", which are computational Grids dealing with sporadically available resources such as a computer at a beamline or a computer donated for a short period of time to a compute cluster. The InfoGram service can enable a SETI@home type of service, which can be used to integrate machines running on a cluster of MS Windows machines. Besides executing

```
<implMap>
      <mapping>
            <source portName="CMCSPortType" operation="qEngine" />
            <target command="/bin/QEngine" />
      </mapping>
      <mapping>
            <source portName="CMCSPortType" operation="polyFit" />
            <target command="/bin/PolyFit" />
      </mapping>
</implMap>
```

Figure 4. XML mapping file for the command to Web services converter

processes outside of the JVM, we have enhanced the security model for Grid computing while reusing Java's security model to, for example, restrict access to machine resource and prevent Trojan programs.

5.2. Web Services

The Web services approach is quickly gaining popularity in the industry. Web services are designed to provide application integration via the use of standard mechanisms to describe, publish, discover, invoke, and compose themselves. Moreover, Web services are platform and implementation independent. In other words, Web services written in a certain language can be accessed and invoked by clients written in other languages, executing under different environments. This capability is highly appealing to the scientific community, as it enables a high level of collaboration between various pieces of software written by different organizations in different languages. Despite all the advantages of the Web Service technology, currently there are only limited Web service development environments, especially in languages other than Java. In such a scenario, it would be very convenient if there existed a tool that would be able to wrap an existing scientific application and expose it as a Web service. We are exploring the viability of this idea, using a prototypical implementation of a command to Web service converter. This converter is built by using Apache Axis [16] as the development environment. The converter takes as input the service description in the form of a WSDL document as well as an XML-encoded mapping between the operations exported in the WSDL and the target executables that they map to. The converter generates client- and server-side code for the target Web service using the standard Axis WSDL2Java converter, as well as the code for the actual implementation of the Web service using the XML based mapping that has been provided.

An example of the mapping, which has been used as part of the CMCS project [62], is shown in the Figure 4. The qEngine operation maps to the executable "/bin/QEngine", while the polyFit operation maps to the executable "/bin/PolyFit". The scientific codes can then be converted into Web services by automatic generation of wrapper code using the information defined in XML format. These Web services can then be deployed, so that remote clients can have access to these codes over the network. We are currently analyzing patterns that would be appropriate for code generation. Such patterns have to be suitably captured in the XML mapfile and understood by the code generator so as to generate appropriate glue code.

6. ADVANCED CoG Kit COMPONENTS

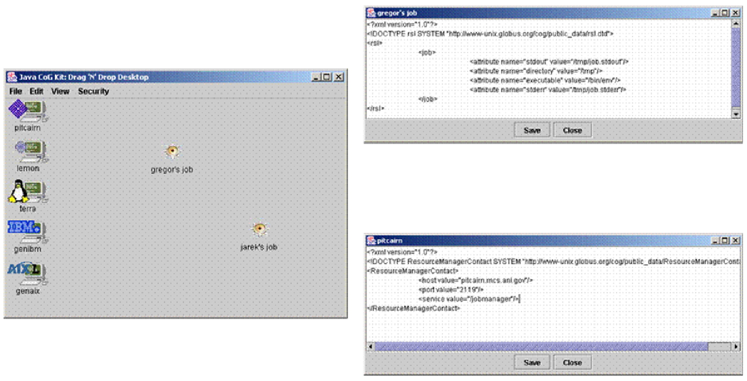
Now that we have illustrated the usefulness of CoG Kits, using the example of the Java CoG Kit, we demonstrate how we use it to provide clients with access to advanced services to clients. As we have seen in Figure 2, we desire to implement services related to job, data, and workflow management. We have developed prototypes of advanced services and client interfaces that address these issues. Together these components can be used as part of a GCE. Other suggestions for components and services are listed in [60] and [62].

6.1. Sample Components

The first component models a desktop in which the user can create job specifications and machine representations through simple icons. Dragging a job onto a machine will automatically start the execution of this job on the remote machine. The user is able to monitor all jobs submitted to a particular machine by double-clicking on the machine icon. The associated output of the remote job can be downloaded by clicking on the appropriate file descriptor in the monitoring component. The specification of the icons and the associations to jobs and machines are represented in XML format. Figure 5 shows a screenshot of this component.

The second component is an interface to file transfers based on various protocols such as ftp [18], gsiftp [20], gridftp [19], and Reliable File Transfer (RFT) [21]. It is a drag-and-drop component allowing the user to conveniently use third-party file transfers between different Globus ftp servers by using either the gridftp or the gsiftp protocols. While using the RFT protocol, the user can also monitor the progress of reliable file transfers which are executing in parallel. Figure 6 shows the snapshot for this component.

The third component is a workflow component that is currently used to define the workflow of an application in a graphical fashion, with the possibility to define dependencies between tasks as a hypergraph while using a graph data structure in recursive fashion. This feature allows the user to conveniently define large graphs hierarchically, thus increasing the readability. Such a tool could also be modified to create graph representations used by other projects such as Condor-G [36] and OGSA [6] while specifying dependencies between Grid Services. Therefore, the usefulness of such a component goes beyond the simple use as part of a dependency graph creation for simple job executions. Figure 7 shows how a workflow can be defined using this component.



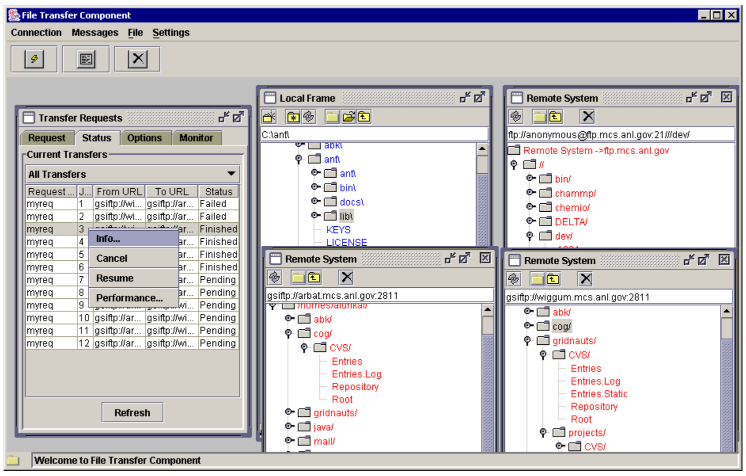
gregor's job

```
<?xml version="1.0"?>
<!DOCTYPE rsl SYSTEM "http://www-unix.globus.org/cog/public_data/rsl.dtd">
<rsl>
        <job>
                <attribute name="stdout" value="/tmp/job.stdout"/>
                <attribute name="directory" value="/tmp"/>
                <attribute name="executable" value="/bin/env"/>
                <attribute name="stderr" value="/tmp/job.stderr"/>
        </job>
</rsl>
```

pitcairn

```
<?xml version="1.0"?>
<!DOCTYPE ResourceManagerContact SYSTEM "http://www-unix.globus.org/cog/public_data/ResourceManagerCont
<ResourceManagerContact>
        <host value="pitcairn.mcs.anl.gov"/>
        <port value="2119"/>
        <service value="/jobmanager"/>
</ResourceManagerContact>
```

Figure 5. A prototypical GUI component performing job management for the GCE using the Java CoG Kit.



Request ...	J...	From URL	To URL	Status
myreq	1	gsiftp://wi...	gsiftp://ar...	Failed
myreq	2	gsiftp://wi...	gsiftp://ar...	Failed
myreq	3	[illegible]	gsiftp://ar...	Finished
myreq	4	[illegible]	ar...	Finished
myreq	5	[illegible]	ar...	Finished
myreq	6	[illegible]	ar...	Finished
myreq	7	[illegible]	ar...	Pending
myreq	8	[illegible]	ar...	Pending
myreq	9	[illegible]	wi...	Pending
myreq	10	gsiftp://ar...	gsiftp://wi...	Pending
myreq	11	gsiftp://ar...	gsiftp://wi...	Pending
myreq	12	gsiftp://ar...	gsiftp://wi...	Pending

C:\ant\
ftp://anonymous@ftp.mcs.anl.gov:21///dev/
Remote System ->ftp.mcs.anl.gov
gsiftp://arbat.mcs.anl.gov:2811
gsiftp://wiggum.mcs.anl.gov:2811

Figure 6. A prototypical GUI performing data management for the GCE using the Java CoG Kit.

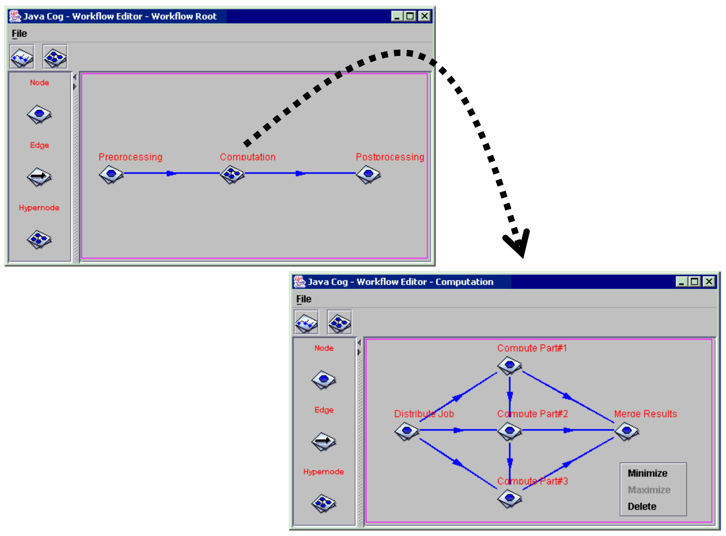


Figure 7. A prototypical component using the Java CoG Kit to perform workflow for the GCE

6.2. Community Use

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;
- *portal developers* to create portals that expose transparently the Grid functionality as part of a portal service; and
- *application developers* to use of Grid services within the application portal.

A subset of projects currently using the Java CoG Kit for accessing Grid functionality includes the following:

- CoGBox [54] provides a simple GUI for much of the client-side functionality such as file transfer and job submission.
- CCAT [9] and XCAT [40] provide an implementation of a standardsuggested bythe 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) [7] 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) [53] is a computing portal system that provides uniform access layer to a large variety of Grid services including other Problem Solving Environments, libraries, and applications.

Table II. Examples of Community CoG Kits

Language	Name	Globus Compatibility	Web Link
Perl	Perl CoG	API based	gridport.npaci.edu/cog/
Python	pyGlobus	API based	www-itg.lbl.gov/gtg/projects/pyGlobus/
Java	Java CoG Kit	protocol based	www.globus.org/cog
JSP	GPDK	through Java CoG Kit	doesciencegrid.org/projects/GPDK/
CORBA	CORBA CoG	through Java CoG Kit	www.caip.rutgers.edu/TASSL/Projects/CorbaCoG/

- The NASA IPG LaunchPad [11] 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 [42] 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) [12] is building a computational collaboratory to bring the numerical treatment of the Einstein Theory of General Relativity to astrophysical problems.
- TENT [37] is a distributed simulation and integration system used, for example, for airplane design in commercial settings.
- ProActive [28] 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 standard Java RMI library. The CoG Kit provides access to the Grid.
- DISCOVER [30] 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 and Java Commodity Grid Kits.
- The Java CORBA CoG Kit [17] provides a simple Grid domain that can be accessed from CORBA clients. Future implementations in C++ are possible.
- The UNICORE [14] project as part of the Grid Interoperability Project (GRIP) [3] uses the Java CoG Kit to interface with Globus.

Additionally, work is currently performed as part of the Globus Project to provide a reference implementation of the Open Grid Service Architecture (OGSA) proposed through the Global Grid Forum. The current technology preview uses the Java CoG Kit's GSI security implementation and a modified version of the Java CoG Kit's GRAM gatekeeper. The role of the Java CoG Kit for some of these projects is depicted in Figure 8.

A regularly updated list of such projects can be found at http://www.cogkits.org. We encourage the users to notify us of additional projects using CoG Kits, so that we can receive feedback about the requirements of the community. We like also to document the use and existence of other CoG Kits. In Table II we list a number of successfully used CoG Kits.

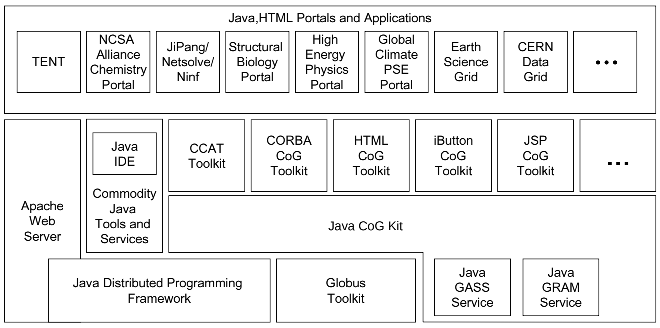


Figure 8. The Java CoG Kit builds a solid foundation for developing Grid applications based on the ability to combine Grid and Web technologies.

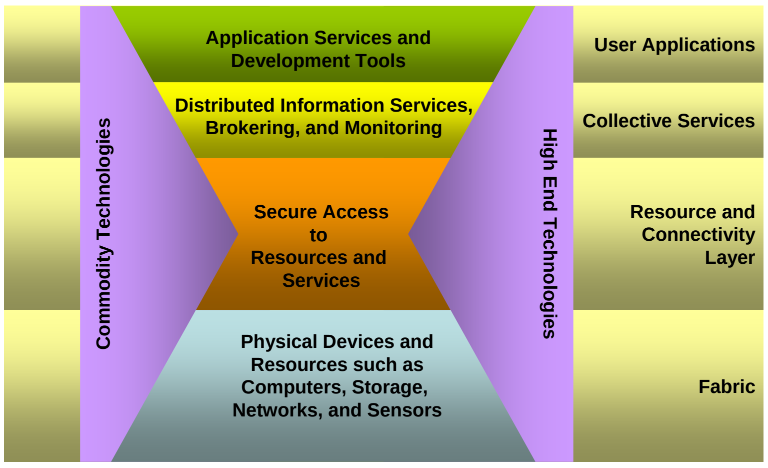

Application Services and Development Tools
Distributed Information Services, Brokering, and Monitoring
Secure Access to Resources and Services
Physical Devices and Resources such as Computers, Storage, Networks, and Sensors
Commodity Technologies
High End Technologies
User Applications
Collective Services
Resource and Connectivity Layer
Fabric

Figure 9. Commodity and high-end technologies bring enhanced value to the core Grid architecture.

7. 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 technologies and development environments. Various Commodity Grid projects are creating such a bridge for different commodity technologies. As part of the Java and Python Commodity Grid project we provide an elementary set of classes that allow the Java and Python programmers to access basic Grid services, as well as enhanced services suitable for the definition of desktop problem solving environments. Additionally, we provided the Globus Toolkit with an independent set of client tools that was able to increase the code quality of the C Globus Toolkit and the productivity of the end user.

Our future work will involve the integration of more advanced services into the Java CoG Kit and the creation of other CoG Kits and the integration of Web services technologies. We hope to gain a better understanding of where changes to commodity or Grid technologies can facilitate interoperability and how commodity technologies can be exploited in Grid environments. We believe that it is important to develop middleware for creating Grid Computing Environments. We emphasize that a CoG Kit provides more than just an API to existing Grid services. Indeed, it brings the modalities and the unique strength of the appropriate commodity technology to the Grid as the Grid brings its unique strengths to the commodity users. This relationship is summarized in Figure 9, the modified the Grid architecture [32] which is introduced in [61] with the explicit vertical support for a variety of commodity and high-end technologies into the Grid architecture.

8. AVAILABILITY

The Java CoG Kit closely monitors the development within the Globus Project to ensure that interoperability is maintained. The CoG Kit development team continues to keep track of projects that use 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. For up-to-date release notes, readers should refer to the Web page at http://www.globus.org/cog, where the Java CoG kit is available for download. New releases are announced to the mailing list at cog-news@globus.org. Information about other CoG Kits such as Python, Perl, and CORBA can also be obtained from this Web page. We welcome contributions and feedback from the community.

9. ACKNOWLEDGEMENTS

This work is 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 supported Globus Toolkit research and development. We thank Ian Foster, Keith Jackson, Geoffrey C. Fox, Dennis Gannon, Shawn Hampton, Manish Parashar, Snigdha Verma, Mary Thomas, and Jay Alameda for the valuable discussions during the course of the ongoing CoG Kit development. We thank Nell Rehn, Peter Lane, Pawel Plaszczak, Mike Russell, Jason Novotny, Ravi Madduri, Benjamin Temko, Shamsuddin Ladha, Beulah Alunkal, and all others who have contributed to the Java CoG Kit. We thank Gail Pieper for her valuable comments. This work would not have been possible without the help of the Globus Project team. The Java CoG Kit is developed at Argonne National Laboratory as part of the Globus Project. Globus Toolkit and Globus Project are trademarks held by the University of Chicago.

REFERENCES

1. Global Grid Forum. www.gridforum.org.
2. Global Grid Forum Grid Computing Environments Working Group. www.computingportals.org.
3. Grid Interoperability Project. http://www.grid-interoperability.org/.
4. Grid Security Infrastructure. http://www.globus.org/security.
5. Java Grande Forum. www.javagrande.org.
6. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. http://www.globus.org/research/papers/ogsa.pdf.
7. The Grid Portal Development Kit, 2000. http://dast.nlanr.net/Projects/GridPortal/.
8. iButton Web Page, 2001. http://www.ibutton.com/.
9. Indiana CCAT Home Page, 2001. http://www.extreme.indiana.edu/ccat/.
10. Java Web Start, Version 1.0.1 Edition, 2001. http://www.sun.com/products/javawebstart/.
11. Launching into Grid Space with the NASA IPG Launchpad, 2001. http://www.nas.nasa.gov/Main/Features/2001/Winter/launchpad.html.
12. The Astrophysics Simulation Collaboratory: A Laboratory for Large Scale Simulation of Relativistic Astrophysics, 2001. http://www.ascportal.org/.
13. The Globus Project WWW Page, 2001. http://www.globus.org/.
14. UNICORE. 2001. http://www.unicore.de/.
15. XML Schema, Primer 0 - 3, 2001. http://www.w3.org/XML/Schema.
16. Apache Axis, 2002. http://xml.apache.org/axis/.
17. CORBA CoG Kits, 2002. http://www.globus.org/cog/corba/index.html/.
18. File Transfer Protocol, 2002. http://www.w3.org/Protocols/rfc959/Overview.html.
19. Grid FTP, 2002. http://www.globus.org/datagrid/gridftp.html.
20. GSI Enabled FTP, 2002. http://www.globus.org/datagrid/deliverables/gsiftp-tools.html.
21. Reliable File Transfer Service, 2002. http://www-unix.mcs.anl.gov/ madduri/RFT.html.
22. Resource Specification Language, 2002. http://www.globus.org/gram/gram_rsl_parameters.html.
23. W. Allcock, A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, and S. Tuecke. The data grid: Towards an architecture for the distributed management and analysis of large scientific datasets. *Journal of Network and Computer Applications*, 23:187–200, 2001.
24. D.M. Beazley. Using SWIG to control, prototype, and debug C program with Python. 1997 (to appear).
25. D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. Nielsen, S. Thatte, and D. Winer. Simple Object Access Protocol (SOAP) 1.1, 2000. http://www.w3.org/TR/SOAP.
26. E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. Web Services Description Language (WSDL) 1.1, 2001. http://www.w3.org/TR/wsdl.
27. K. Czajkowski, I. Foster, and C. Kesselman. Resource co-allocation in Computational Grids, 1999.
28. D. Caromel. ProActive Java Library for Parallel, Distributed and Concurrent Programming, 2001. http://www-sop.inria.fr/oasis/ProActive/.
29. W. Edwards. *Core JINI, 2nd edition.* Prentice Hall, 2000.
30. M. Parashar et al. DISCOVER, 2001. http://www.discoverportal.org/.

31. I. Foster, C. Kesselman, and S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International Journal of Supercomputer Applications*, 15(3), 2001. http://www.globus.org/research/papers/anatomy.pdf.
32. Ian Foster. The Grid: A New Infrastructure for 21st Century Science. *Physics Today*, 55(22):42, 2002. http://www.aip.org/pt/vol-55/iss-2/p42.html.
33. Ian Foster, Joeseph Insleay, Gregor von Laszewski, Carl Kesselman, and Marcus Thiebaux. Data Visualization: Data Exploration on the Grid. *IEEE Computer*, 14:36–41, Dec. 1999.
34. Geoffrey C Fox. Portals for Web Based Education and Computational Science, 2000.
35. Geoffrey C. Fox and Wojtek Furmanski. High Performance Commodity Computing. In Ian Foster and Carl Kesselman, editors, *The Grid: Bluepirnt for a new computing infrastructure*. Morgam Kaufman, 1999.
36. James Frey, Todd Tannenbaum, Ian Foster, Miron Livny, and Steven Tuecke. Condor-G: A Computation Management Agent for Multi-Institutional Grids. In *Proceedings of the Tenth IEEE Symposium on High Performance Distributed Computing (HPDC10)*, San Francisco, 2001. IEEE Press.
37. German Air and Space Agency (DLR). TENT Home Page, 2001. http://www.sistec.dlr.de/tent/.
38. Vladimir Getov, Gregor von Laszewski, Michael Philippsen, and Ian Foster. Multi-Paradigm Communications in Java for Grid Computing. *Communications of ACM*, 44(10):119–125, October 2001. http://www.globus.org/cog/documentataion/papers/.
39. Li Gong. Get connected with Jxta. Sun MicroSystems, Java One, June 2001.
40. M. Govindaraju, S. Krishnan, K. Chiu, A. Slominski, D. Gannon, and R. Bramley. XCAT 2.0 : A Component Based Programming Model for Grid Web Services. In *Submitted to Grid 2002, 3rd International Workshop on Grid Computing*, 2002.
41. Keith Jackson. pyGlobus - A CoG Kit for Python. *to be published*, 2002.
42. 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. In *Proceedings of SC2001*, 2001.
43. M. Hall. *Core Servlets and JavaServer Pages (JSP), 1st Edition.* Prentice Hall / Sun Microsystems Press, 2000.
44. Richard Monson-Haefel. *Enterprise JavaBeans.* O'Reilly, 3rd edition, October 2001.
45. J. Novotny, S. Tuecke, and V. Welch. An Online Credential Repository for the Grid: MyProxy . In *Proceedings of the Tenth International Symposium on High Performance Distributed Computing (HPDC-10)*, San Francisco, August 2001. IEEE Press.
46. Jason Novotny. The Grid Portal Development Kit. *to be published*, 2001. http://dast.nlanr.net/Features/GridPortal/.
47. R. Lee. Jndi api tutorial and reference: Building directory-enabled java applications, 2000. Addison-Wesley.
48. Michael Russell, Gabrielle Allen, Ian Foster, Ed Seidel, Jason Novotny, John Shalf, Gregor von Laszewski, and Greg Daues. The Astrophysics Simulation Collaboratory: A Science Portal Enabling Community Software Development. *Journal on Cluster Computing*, 5(3):297–304, July 2002.
49. S. Holzner. *Inside XML, 1st Edition.* New Riders Publishing, 2000.
50. J. Siegel. *Corba 3 : Fundamentals and Programming, 2nd ed.* John Wiley and Sons, 2000.
51. Larry Smarr. Infrastructures for Science Portals, 2001. http://www.computer.org/internet/v4n1/smarr.htm.
52. Sun Microsystems. Java Authentication and Authorization Service (JAAS), 2001. http://java.sun.com/products/jaas/.
53. T. Suzumura, S. Matsuoka, and H. Nakada. A Jini-based Computing Portal System, 2001. http://matsu-www.is.titech.ac.jp/ suzumura/jipang/.
54. B. Temko. The CoGBox Home Page. http://www.extreme.indiana.edu/ btemko/cogbox/.
55. Mary Thomas, Steve Mock, and Gregor von Laszewski. A Perl Commodity Grid Kit. *Concurrency and Computation: Practice and Experience*, accepted.
56. Snighda Verma, Manish Parashar, Jarek Gawor, and Gregor von Laszewski. Design and Implementation of a CORBA Commodity Grid Kit. In Craig A. Lee, editor, *Second International Workshop on Grid Computing - GRID 2001*, number 2241 in Lecture Notes in Computer Science, pages 2–12, Denver, November 2001. In conjunction with SC'01, Springer. http://www.caip.rutgers.edu/TASSL/CorbaCoG/CORBACog.htm.
57. Gregor von Laszewski, Eric Blau, Michael Bletzinger, Jarek Gawor, Peter Lane, Stuart Martin, and Michael Russell. Software, Component, and Service Deployment in Computational Grids. In Judith Bishop, editor, *IFIP/ACM Working Conference on Component Deployment*, volume 2370 of *Lecture Notes in Computer Science*, pages 244–256, Berlin, Germany, June 20-21 2002. Springer. http://www.globus.org/cog.

58. Gregor von Laszewski, S. Fitzgerald, I. Foster, C. Kesselman, W. Smith, and S. Tuecke. A Directory Service for Configuring High-Performance Distributed Computations. In *Proceedings of the 6th IEEE Symposium on High-Performance Distributed Computing*, pages 365–375, August 5-8 1997.
59. Gregor von Laszewski, Ian Foster, Jarek Gawor, and Peter Lane. A Java Commodity Grid Kit. *Concurrency and Computation: Practice and Experience*, 13(8-9):643–662, 2001. http://www.globus.org/cog/documentation/papers/cog-cpe-final.pdf.
60. Gregor von Laszewski, Ian Foster, Jarek Gawor, Peter Lane, Nell Rehn, and Mike Russell. Designing Grid-based Problem Solving Environments and Portals. In *34th Hawaiian International Conference on System Science*, Maui, Hawaii, January 3-6 2001. http://www.mcs.anl.gov/ laszewsk/papers/cog-pse-final.pdf, http://computer.org/Proceedings/hicss/0981/volume
61. Gregor von Laszewski, Gail Pieper, and Patrick Wagstrom. *Performance Evaluation and Characterization of Parallel and Distributed Computing Tools*, chapter Gestalt of the Grid. Wiley Book Series on Parallel and Distributed Computing. to be published.
62. Gregor von Laszewski, Branko Ruscic, Patrick Wagstrom, Sriram Krishnan, Kaizar Amin, Reinhardt Pinzon, Melita L. Morton, Sandra Bittner, Mike Minkoff, Al Wagner, and John C. Hewson. A Grid Service Based Active Thermochemical Table Framework. In *Preprint ANL/MCS-P972-0702*, 2002.
63. Gregor von Laszewski, Mary Westbrook, Ian Foster, Edwin Westbrook, and Craig Barnes. Using Computational Grid Capabilities to Enhance the Ability of an X-Ray Source for Structural Biology. *Cluster Computing*, 3(3):187–199, 2000. ftp://info.mcs.anl.gov/pub/tech_reports/P785.ps.Z.
64. Gregor von Laszewski Jarek Gawor Carlos J. Peña and Ian Foster. InfoGram: A Peer-to-Peer Information and Job Submission Service. In *Proceedings of the 11th Symposium on High Performance Distributed Computing*, Edinburgh, Scotland, July 24-26 2002. http://www.globus.org/cog.