GRID INFRASTRUCTURE TO SUPPORT SCIENCE PORTALS FOR LARGE SCALE INSTRUMENTS

GREGOR VON LASZEWSKI AND IAN FOSTER

Mathematics and Computer Science Division
Argonne National Laboratory
9700 S. Cass Avenue
Argonne, IL 60439, U.S.A.
gregor@mcs.anl.gov

Abstract

Soon, a new generation of scientific workbenches will be developed as a collaborative effort among various research institutions in the United States. These scientific workbenches will be accesses in the Web via portals. Reusable components are needed to build such portals for different scientific disciplines, allowing uniform desktop access to remote resources. Such components will include tools and services enabling easy collaboration, job submission, job monitoring, component discovery, and persistent object storage. Based on experience gained from Grand Challenge applications for large-scale instruments, we demonstrate how Grid infrastructure components can be used to support the implementation of science portals. The availability of these components will simplify the prototype implementation of a common portal architecture.

Keywords: *Science Portal, Grid, Metacomputing, Globus*

1. Introduction

A driving force for developing the next generation of computing infrastructure is scientific applications. It is important to identify common components that are neede and can be used by different application areas [28]. The availability of common and reusable components will accelerate and enhance the development of similar efforts in other application areas. In this paper, we report on lessons learned during the development of programs for Grand Challenge applications as part of large scale data analysis and instrumentation in computed microtomography (CMT) and structural biology (SB). We believe that many of the issues addressed in these disciplines are typical for other scientific problems.
Once the common components between different disciplines are identified, they will be used to build *science portals* [14]. In general, a science portal defines a gateway or portal to information related to a particular scientific discipline or area of research. The portal serves as Web interface to tools, databases, and other useful services including job

submission and collaborative spaces. Science portals are a continuation of the successfully implemented scientific workbenches [4]. Unfortunately, the current generation of available tools used to construct workbenches is hard to maintain. Since only a limited shared infrastructure exists, it is not easy to derive ports from one discipline to another. New efforts will need fundamentally different technologies. Naturally, portals for various scientific disciplines might look quite different. Nevertheless, many of them will share a number of common toolkit components.
The paper is structured as follows. First, we outline some basic scientific and computational challenges of the computed microtomography and structural biology projects. Analysis of the workflow of these experiments helps to establish a common set of components that must be supported to ease the work of the scientist. The demand for implementation of Webbased science portals is supported by selected Grid services, which are then discussed. We point out how to utilize XML- and LDAP-based formats to allow interoperability of components developed in a language-neutral and compute-framework-neutral way. We conclude the paper with a summary of the current state of the project and point out opportunities for further research.

2. Computed Microtomography and Structural Biology

Before we can describe the general architecture of our approach, we present some application-specific details that help to explain the goals and limitations of the project in the light of developing a science portal.
In the areas of CMT and SB, the use of x-rays as a nondestructive tool for investigating the *internal* structure of materials at the micron length scale has grown rapidly over the past decade as a result of the advent of synchrotron radiation sources. In a typical CMT or SB experiment, a sample is illuminated by a beam of x-rays, and data is collected for multiple sample orientations by using a charge-coupled device (CCD) [33, 1]. These images are then used to derive a three-dimensional representation of the analyzed objects. This data contains quantitative information about the x-ray attenuation coefficient at a particular x-ray energy. However, the effective use of such an expensive instrument requires the ability to archive, analyze, and visualize the collected data at orders of magnitude more than is currently possible. The data rates and compute power required to address this Grand Challenge problem are prodigious, easily reaching one gigabit per second and a teraflop per second.
We illustrate this statement with a scenario found during the reconstruction of microtomographic data, where a time-consuming reconstruction process is used to obtain a three-dimensional raw data set representing with spatial resolution of as little as 1 μm. A 3-D raw data set generated by a typical detector will comprise 1000 1024×1500 two-byte slices (3 GB); detectors with significantly higher resolutions will soon be available. If we assume current reconstruction techniques and make fairly optimistic scaling assumptions, reconstruction of this dataset requires about 10^{13} floating-point operations (10 Tflops). On a 100 Mflop/sec workstation, this translates to 32 hours; on a 1 Tflop/sec computer, it

would take about 10 seconds. With current detector technologies, this dataset might take 1500 seconds to acquire; however, new detectors will improve readout times considerably [29].
Even larger estimations apply to the discipline of structural biology, where the data gathered during such experiments is used to determine the molecular structure of macromolecules to enhance, for example, the capabilities of modern drug design in the fields of basic and applied research [30]. The total amount of data in many of the experiments we conducted was 20 GB. A considerable amount of postprocessing has to be performed following the data acquisition phase to obtain the structural solution of the molecular sample. For small structures, the solution usually does not pose a problem. For macromolecules, the computational demand on even the fastest processors can be months. This is due to the enormous and irregular solution space for which it is currently difficult to write fast deterministic algorithms to obtain an expectable solution.
The many orders of magnitude increase in brilliance now available at third-generation sources such as the Advanced Photon Source (APS) allows dramatic improvements in temporal resolution. In addition, the availability of powerful computational grids will make it feasible to obtain a 3-D representation in a reasonable response time. The requirements of these photon source applications can be fulfilled with the infrastructure provided by a computational grid [10]. The term "grid" is chosen in analogy with the electric power grid, which provides pervasive access to power and, like the computer and a small number of other advances, had a dramatic impact on human capabilities. By providing pervasive, dependable, and consistent access to advanced computational capabilities, the application of supercomputing-enhanced photon source algorithms allows real-time operation during an experiment while using nonlocal computational resources to allow a certain level of quality of service.

3. Analysis of Application-Specific Requirements

It is beyond the scope of this paper to outline the contents of a science portal for large-scale instruments. This effort will require considerable resources and commitment by experts in order to derive a contents base that appeals to many users [32, 2, 31]. Nevertheless, it is possible to derive the requirements for tools that will allow the simplification for the construction of such portals. In the following sections we outline these basic requirements and point to potential solutions provided as part of the Grid infrastructure.
To extract common modalities, we outline the traditional model to perform a photon source experiment:

1. Gather knowledge, and perform research to develop an experiment.
2. Plan the experiment.
3. Prepare the samples and the hardware at the beamline.
4. Perform the data acquisition (e.g., collect the images).

5. Select a reconstruction algorithm.

6. Choose parameters for the reconstruction algorithm.

7. Perform the actual reconstruction.

8. Visualize the reconstructed data.

9. Go to step 5, or, if no parameters can be found, check whether the experiment was correct, or develop new algorithms and go to step 4 if necessary.

From these basic steps it is clear that the following components must be part of a science portal for large photon sources:

- Access to databases for planning the experiments and conducting research.
- Access to the computational Grid for execution of the compute-intensive task under deadline constraints.
- Access to a uniform collaborative environment that allows the exchange of ideas in various formats, including, text, drawings, speech, and sophisticated immersive visualization.
- Access to a software repository to store new components.
- Access to a parameter repository to store different configurations for the experiment parameters.

The computational steps currently performed by the researchers feature hands-on use and batch processing. A major difficulty with the current practice is the turnaround time between the data acquisition and the reconstruction, often due to lack of available computing power. This is especially problematic for synchrotron-based experiments because only a limited amount of expensive beam time is available for a user. The use of advanced reservation-based compute power can reduce this turnaround time to a few hours or minutes, as demonstrated at the SC'98 conference in Florida. Allowing the users to view the results in rapid response time (quasi-real time) gives an opportunity to alter experiment conditions on the fly [22]. This functionalty greatly improves the capabilities of a synchrotron radiation facility.
To characterize the experiment requirements more precisely, we identify two experiment modes: (1) the use of an online operation or experiment as described above and (2) the use of a postprocessing mode, which reconstructs the dataset with a varying parameter set (see Table 1). If enough computational power is available, recalculation can also be performed during an experiment. To enable fast processing, the data must be shipped from the acquisition hardware to the computer performing the reconstruction, preferably connected to large storage media with fast access. At the same time it is important to enable an archival service for the experimentalist in order to allow for data recovery in the case of data loss. As pointed out previously, beam time is limited, and the goal for an experimentalist is to achieve the most progress in this limited time. A fast reconstruction

algorithm can be used to help decide whether the current experiment has to be interrupted prematurely because of an error in the setup. This will allow for an increase in the number of experiments to be conducted per hour. In order to handle the complicated and diverse supercomputing environments, it is essential to provide a simple interface giving the beamline experimentalist control over the parameter set, as well as the possibility to terminate the current calculation at any time.

Besides the requirements driven by the computational aspect of the application, organizational aspects benefit from a remote operational mode. Because of the hazardous and often unpleasant environment, remote operation is desirable. With remote operation, the facility can maintain a small but well-trained team of beamline staff experimentalists. This approach offers several advantages. It reduces the operational and user-specific cost and minimizes travel cost to the unique facility. It allows new user groups to gain access to a unique facility, such as the APS. Furthermore, it increases the access time to the beamline while minimizing the effort required by trained experts to set up experiments.

With the availability of a collaborative and remote steering environment as part of a science portal, new user communities in commercial and educational facilities are likely to use the supercomputing-enhanced light sources in remote fashion. Multiple users at geographically dispersed locations should be able to collaborate easily with each other. A "plug-and-play" environment, as shown in Figure 1, makes it possible for authorized participants in different geographical locations with different visualization engines to attend a collaborative experiment session.

Furthermore, the implementation of the Grid-enabled photon source applications must deal with issues common to metacomputing environments. These issues include the following:

- *Multiple administrative domains.* The resources used by the application often are not owned or administered by a single entity. The need to deal with multiple administrative entities complicates the already challenging network security problem, as different entities may use different authentication mechanisms, authorization schemes, and access policies. The need to execute user-supplied codes at the geographical disperse sites introduces additional concerns.

- *Restriction on the locality of the compute platform.* Many of the tools used by the applications are of commercial nature or cab be run are only on a selected set of machines.

- *Heterogeneity at multiple levels.* Both the computing resources used to construct a virtual computer and the networks that connect these resources are highly heterogeneous. Heterogeneity arises at multiple levels, ranging from physical devices, through system software, to scheduling and usage policies.

- *Unpredictable structure.* Traditionally, high-performance applications have been developed for a single class of systems with well-known characteristics—or even for one particular computer. Geographical distribution and complexity are other factors that make it difficult to determine system characteristics such as network bandwidth and latency *a priori.*

Table 1: The modes in which an experiment on the advanced photon source is executed pose different demand on the compute environment.

	experiment mode	*postprocessing mode*
speed	finish by deadline (end of experiment)	as fast as possible (quasi interactive mode)
-“-	as fast as possible interrupt if wrong results	over “lunchbreak”
space	secure, fast, large, backup	keep up with calculation speed, read from backup
parameters	use standard parameters	modifying the parameter set to its optimum
-“-	run postprocessing mode at experiment time	

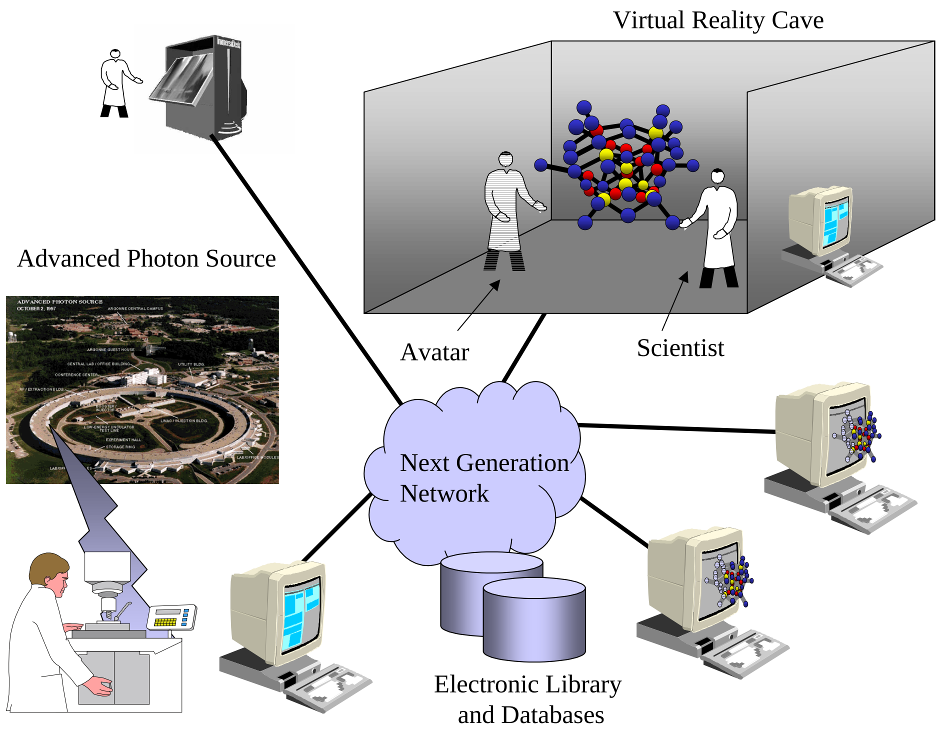


Figure 1: The “grid-enabled” photon source allows researchers to display the same state of the visualized object on all display stations participating in a collaborative session [30]. Remote computation and steering become possible.

- *Dynamic and unpredictable behavior.* Traditional high-performance systems use scheduling strategies such as space sharing or gang-scheduling to provide exclusive—and hence predictable—access to processors and networks. In metacomputing environments, resources—especially networks—are more likely to be shared. One consequence of sharing is that behavior and performance can vary over time. For example, in wide area networks built using the Internet Protocol suite, network characteristics such as latency, bandwidth, and jitter may vary as traffic is rerouted. Large-scale metasystems may also suffer from network and resource failures. In general, it is a challenging problem to guarantee even minimum quality-of-service requirements.

Fundamental to all of these issues is the need for mechanisms that allow applications to obtain real-time information about system structure and state, use that information to make configuration decisions, and be notified when information changes. Required information can include network activity, available network interfaces, processor characteristics, and authentication mechanisms. Decision processes can require complex combinations of these data in order to achieve efficient end-to-end configuration of complex networked systems.

4. Enabling Grid Middleware Components

Central to the development of such widely distributed applications are middleware services that enable resource discovery, user authentication and authorization, resource scheduling and execution control. A number of pioneering efforts have produced useful services for the metacomputing application developer. For example, Parallel Virtual Machine (PVM) [15] and the Message Passing Interface (MPI) [17] provide a machine-independent communication layer, Condor [19] provides a uniform view of processor resources, Legion [16] builds system components on a distributed object-oriented model, and the Andrew File System (AFS) [21] provides a uniform view of file resources. Each of these systems has been proven effective in large-scale application experiments.
The Globus metacomputing toolkit provides middleware services necessary to construct a computational grid infrastructure and to develop applications that use grid-based supercomputers, mass storage, and immersive visualization facilities. Our goal in the Globus project is not to compete with these and other related efforts, but rather to provide basic infrastructure that can be used to construct portable, high performance implementations of a range of such services. To this end, we focus on (a) the development of low-level mechanisms that can be used to implement higher-level services, and (b) techniques that allow those services to observe and guide the operation of these mechanisms. If successful, this approach can reduce the complexity and improve the quality of metacomputing software by allowing a single low-level infrastructure to be used for many purposes and by providing solutions to the configuration problem in metacomputing systems.
To demonstrate that the Globus approach is workable, we must show that it is possible to use a single set of low-level mechanisms to construct efficient implementations of diverse services on multiple platforms and show their usability in application programs.

5. Globus Toolkit Modules

The Globus toolkit comprises a set of modules and services. Each module defines an *interface*, which higher-level services use to invoke that module's mechanisms, and provides an *implementation*, which uses appropriate low-level operations to implement these mechanisms in different environments.
The availability of these modules will help to define services that are important for many scientific applications:

- *Resource location and allocation.* This component provides mechanisms for expressing application resource requirements, for identifying resources that meet these requirements, and for scheduling resources once they have been located. Resource location mechanisms are required because applications cannot, in general, be expected to know the exact location of required resources, particularly when load and resource availability can vary. Resource allocation involves scheduling the resource and performing any initialization required for subsequent process creation, data access, and so forth. In some situations—for example, on some supercomputers—location and allocation must be performed in a single step [9]. In science portals Resource management services must be built to allow specification of what computing, network, and storage are needed by an application.

- *Authentication interface.* This component provides basic authentication mechanisms that can be used to validate the identity of both users and resources. These mechanisms provide building blocks for other security services such as authorization and data security that need to know the identity of parties engaged in an operation [12]. A security service with a single sign-on using public key infrastructure for all resources must be available in a science portal.

- *Unified resource information service.* This component provides a uniform mechanism for obtaining real-time information about metasystem structure and status. The mechanism allows components to post as well as receive information. Scoping and access control are also supported[7].

- *Data access.* This component is provides high-speed remote access to persistent storage such as files. Globus contains a remote data access service via URLs [3].

- *Communications.* This component provides basic communication mechanisms. These mechanisms permit the efficient implementation of a wide range of communication methods, including message passing, remote procedure call, distributed shared memory, stream-based, and multicast. The mechanisms must be aware of network quality-of-service parameters such as jitter, reliability, latency, and bandwidth [11].

- *Process creation.* This component initiates computation on a resource once it has been located and allocated. This task includes setting up executables, creating an execution environment, starting an executable, passing arguments, integrating the new process into the rest of the computation, and managing termination and process shutdown [5].

- *Process Monitoring.* This component reports on the status of a compute resource or task. A detection of failure allows one to develop a fault-tolerant machine state and job state service [24].

Together, the various Globus toolkit modules can be thought of as defining a *metacomputing virtual machine.* The definition of this virtual machine simplifies application development and enhances portability by allowing programmers to think of geographically distributed, heterogeneous collections of resources as unified entities. More information about each component can be found in the cited references.

5.1. Support for Resource-Aware Services and Applications

Metacomputing applications often need to operate networking and computing resources at close to maximum performance. Hence, metacomputing environments must allow programmers to observe differences in system resource characteristics and to guide how these resources are used to implement higher-level services. Achieving these goals without compromising portability is a significant challenge for the designer of metacomputing software.

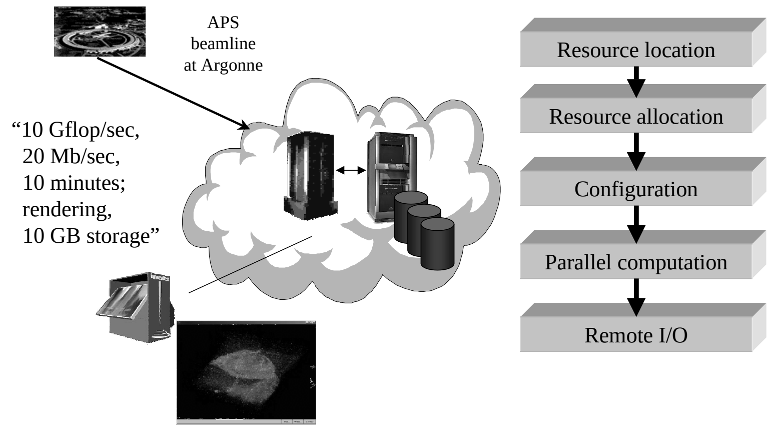


Figure 2: A selection process of resources to fulfill the real-time requirements of the parallel x-ray microtomography program.

We use the Globus communication module to illustrate some of these issues (see Figure 2). This module must select, for each call to its communication functions, one of several low-level mechanisms. On a local-area network, communication might be performed with TCP/IP, while in a parallel computer, specialized high-performance protocols typically offer higher bandwidth and lower latencies. In a wide area environment, specialized ATM protocols can be more efficient. The ability to manage protocol parameters (TCP

packet size, network quality of service) further complicates the picture. The choice of low-level mechanism for a particular communication is a nontrivial problem that can have significant implications for application performance.
Globus toolkit modules address this problem by providing interfaces that allow the selection process to be exposed to, and guided by, higher-level tools and applications. These interfaces provide rule-based *selection*, resource property *inquiry*, and *notification* mechanisms.

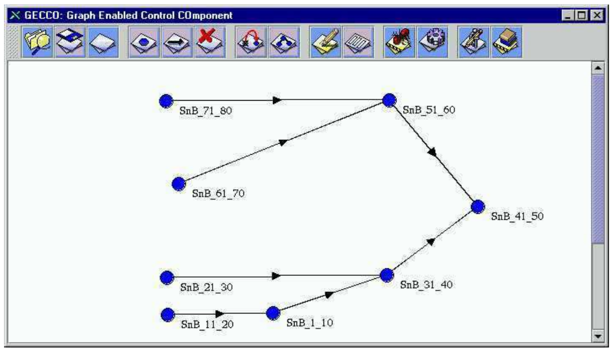
- *Rule-based selection.* Globus modules can identify selection points at which choices from among alternatives (resources, parameter values, etc.) are made. Associated with each selection point is a default selection rule provided by the module developer (e.g., "use TCP packet size X," "use TCP over ATM"). A rule replacement mechanism allows higher-level services to specify alternative strategies ("use TCP packet size Y," "use specialized ATM protocols").
- *Resource property inquiry.* Information provided by the unified information service can be used to guide selection processes within both Globus modules and applications that use these modules. For example, a user might provide a rule that states "use ATM interface if load is low, otherwise Internet," hence using information about network load to guide resource selection.
- *Notification.* A notification mechanism allows a higher-level service or application to specify constraints on the quality of service delivered by a Globus service and to name a call-back function that should be invoked if these constraints are violated. This mechanism can be used, for example, to switch between networks when one becomes loaded.

Higher-level services and applications can use Globus selection, inquiry, and notification mechanisms to *configure* computations efficiently for available resources, and/or to *adapt* behavior when the quantity and/or quality of available resources changes dynamically during execution. For example, consider an application that performs computation on one computer and transfers data over a wide-area network for visualization at remote sites. At startup time, the application can determine available computational power and network capacity and configure its computational and communication structures appropriately (e.g., it might decide to use compression for some data but not others). During execution, notification mechanisms allow it to adapt to changes in network quality of service.
We use the term Adaptive Wide Area Resource Environment (AWARE) to denote a set of application interfaces, higher-level services, and adaptation policies that enable specific classes of applications to exploit a metacomputing environment efficiently. We are investigating AWARE components for several applications, and anticipate developing AWARE toolkits for different classes of metacomputing application.

6. Higher-Level Services

Although Globus provides mechanisms for users to locate, access and manage distributed hardware and software resources, there is still a gap between programming distributed

applications using these APIs and conceptual problem solving at higher levels. Thus it is necessary to provide an intermediate layer that enables access to the services by non-expert applications users. Integrating such higher-level services into a science portal will be key for a successful implementation [20].
Today, several software solutions exist that provide software layers above the basic Grid middleware services. Most prominent are MPICH-G [8] and High Performance C++ [18]. In addition, the Globus team has demonstrated on several occasions that it is possible to build higher-level services that provide even graphical user interfaces to the grid components. Such a service, for example, is the Grid Enabled Console COmponet (GECCO). This tool enables the user to formulate jobs or tasks executing in the Grid environment as a task graph (see Figure 3). GECCO allows tasks to be executed in a fault-tolerant manner according to their specification. This may include the automatic location of a compute resources and prestaging of executables and programs on the dynamical selected machines. GECCO communicates directly with an LDAP server, which can be set up locally by the user in order to allow persistent storage of information. GECCO was developed even before the eXtensible Markup Language (XML) standard was finalized and is based on a similar, but more user-friendly specification language. Nevertheless, we intend to provide a XML-based port to allow easy interoperability with other components. XML is a simplified version of SGML, designed especially for Web documents. It enables designers to include syntactic rules within the document. Other higher-level tools are available or under development, including network monitoring tools.



```
...
<node>
   <name> SnB1 </name>
   <label> SnB_71_80 </label>
   <state> running </state>
   ...
</node>
...
```

Figure 3: On the left side a we show a higher-level tool (GECCO) that executes tasks specified as part of a task graph in fault tolerant manner. The graph presented shows an application of a structure determining task graph. On the right side we show a portion of the XML specification of the information displayed in GECCO.

7. Interoperable Grid Components

One of the problems we face when components and services are contributed by a large and diverse user community is how to maintain their interoperability.

The definition of a well-formulated application interface contributes to the usability of such components. Nevertheless, one has to consider issues related to language and framework independence to increase the exposure of the components. One such effort was started by the DOE2000 Common Component Architecture (CCA) effort. It seeks to develop a software component architecture that enables scientists and engineers to build self-contained software components that can be composed into applications that run either on massively parallel supercomputers or in wide-area distributed computing environments. Recent demonstrations of the proposed CCA API [13] and a distributed high-performance computation and visualization application [23] indicate that these new technologies are appropriate for developing domain-specific Next Generation Internet (NGI) applications such as the Advanced Photon Source macromolecular crystallography science portal. It is especially important that a framework be developed that allows DCOM, CORBA, and JavaBeans to interoperate with CCA distributed components.

We suggest basing this architecture on well-known standards and already available grid components. Hence, we guarantee the support of new technologies in the near and distant future. A central part of the architecture must be a scalable repository that allows storing and locating components and their interfaces. Performance characteristics and machine dependencies should be part of this service. Currently, the Globus MDS provides already a scalable lookup service based on LDAP technology. This service can be extended to include data providers on the base of shell scripts, Jini, JDBC, and COS. A provider is defined in such a way that the data obtained by the provider is translated into LDAP. Furthermore, we suggest formulating the interface descriptions in XML while defining a Document Type Definition (DTD).

The use of syntactic rules within XML documents will allow application developers to build smart clients, in which information is checked against an appropriate DTD before a connection to the server is established and the document is submitted. Such a mechanism is essential to increase scalability because common LDAP implementations perform such tests on the server side rather than the client side. XML parsers available in C and Java will guarantee portability.

Examples for the definition of compute-resource related objects are given in [27]. Basic work for defining standard object representation has been performed as part of the Desktop Assess to Remote Resources (DAtoRR) workshops and will be continued in conjunction with the newly started GridForum activities [26, 6, 25]

A simple scenario (see Figure 4) illustrates an application of interoperable components. Assume an application developer decides to publish a component for reuse by other developers. This component can be published today in the MDS. The publication can be performed either in LDAP or in XML (we use XML in our example). If another user wishes to find out wether a particular component can be executed on a dynamical configurable compute environment, he can issue an appropriate search query. A compile service, which is currently under development by the Globus team, may be used to obtain the final executable which is then staged with the Globus resource managing tools. This approach opens up the possibility of integrating other infrastructure frameworks based on CORBA, DOM, RMI, and Jini, since XML-based interfaces are currently under discussion or development by various vendors.

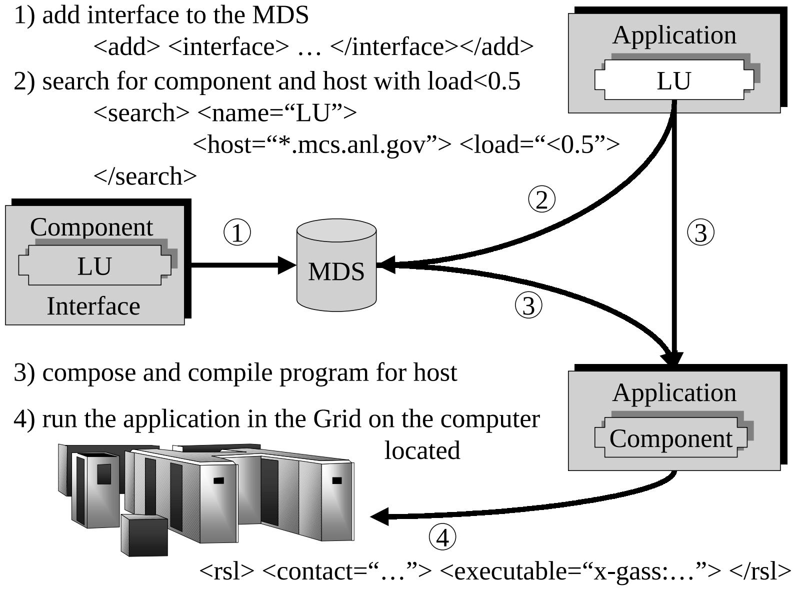

1) add interface to the MDS
<add> <interface> … </interface></add>
2) search for component and host with load<0.5
<search> <name="LU">
<host="*.mcs.anl.gov"> <load="<0.5">
</search>
Application
LU
Component
LU
Interface
MDS
①
②
③
③
3) compose and compile program for host
4) run the application in the Grid on the computer located
Application
Component
④
<rsl> <contact="…"> <executable="x-gass:…"> </rsl>

Figure 4: A component can be assembled for a particular host architecture from its interface definition and the source code, which are placed with compile time instructions into the MDS.

8. Conclusion

In this paper, we have outlined how Grid infrastructure components available today can be applied to develop distributed and resource-aware scientific applications in a heterogeneous compute environment. We used applications at Argonne's Advanced Photon Source as example. The next generation of components developed for the computational Grid must contain features that guarantee the interoperability between components and services developed by various research teams. We propose to use XML-based tools and technologies to formulate and publicize the interfaces. The Metacomputing Directory Service currently provides the possibility to expose such interfaces in a scalable way. Moreover, with XML-based notations, the integration of CORBA and Java-based technology is possible. Where this technology seems not to be sufficient because of the demand on high performance, the Common Component Architecture Group will provide suggestions for alternative and high performance solutions. The work reported here is in its initial phase; prototype implementations will become available soon.

Acknowledgments

This work was supported by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Computational and Technology Research, U.S. De-

partment of Energy, under Contract W-31-109-Eng-38, is supported by DARPA, DOE, and NSF.
We acknowledge the following contributors to this project: Craig Barnes, John Bresnahan, Francesco DeCarlo, Mark Ellisman, Joe Insley, Carl Kesselman, Peter Lane, Derrick C. Mancini, Ian McNulty, James Pflugrath, Mark Rivers, Mei Su, Steve Wang, Mary Westbrook, Ed Westbrook, Steve Young, and all members of the Globus team.

References

[1] National Center for Microscopy and Imaging Research. Web site, ://www-ncmir.ucsd.edu/CMDA/.

[2] The Advanced Photon Source WWW. Web site, http://www.aps.mcs.anl.gov.

[3] J. Bester, I. Foster, C. Kesselman, J. Tedesco, and S. Tuecke. GASS: A Data Movement and Access Service for Wide Area Computing Systems. In *Sixth Workshop on I/O in Parallel and Distributed Systems*, May 1999.

[4] Biological Workbench. Web site, http://biology.ncsa.uiuc.edu/.

[5] K. Czajkowski, I. Foster, N. Karonis, C. Kesselman, S. Martin, W. Smith, and S. Tuecke. A Resource Management Architecture for Metacomputing Systems. In *The 4th Workshop on Job Scheduling Strategies for Parallel Processing*, pages 4–18. IEEE-P, Mar. 1998.

[6] The Datorr Web Site. Web site, ://www-fp.mcs.anl.gov/~gregor/datorr.

[7] S. Fitzgerald, I. Foster, C. Kesselman, G. von Laszewski, W. Smith, and S. Tuecke. A Directory Service for Configuring High Performance Distributed Computations. In *Proc. 6th IEEE Symp. on High Performance Distributed Computing*, pages 365–375. IEEE Computer Society Press, 1997.

[8] I. Foster and N. Karonis. A Grid-Enabled MPI: Message Passing in Heterogeneous Distributed Computing Systems. In *Proc. of the ninth SC Conference*, Nov. 1999.

[9] I. Foster and C. Kesselman. The Globus Project: A Status Report. In *Proc. IPPS/SPDP '98 Heterogeneous Computing Workshop*, pages 4–18, 1998.

[10] I. Foster and C. Kesselman, editors. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufman, July 1998.

[11] I. Foster, C. Kesselman, and S. Tuecke. The Nexus Approach to Integrating Multithreading and Communication. *Journal of Parallel and Distributed Computing*, 37:70–82, 1996.

[12] I. Foster, C. Kesselman, S. Tuecke, and G. Tsudick. A Security Architecture for Computational Grids. In *Proc. of the 5th ACM Conference on Computer and Communication Security*. ACM Press, Nov. 1998.

[13] D. Gannon. A Java Demo of the proposed CCA API. Web site, ://www.extreme.indiana.edu/user/gannon/cca_demo/main.html.

[14] D. Gannon. Gateways to the Access Grid. Internal Draft Document, May 1999.

[15] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, B. Manchek, and V. Sunderam. *PVM: Parallel Virtual Machine—A User's Guide and Tutorial for Network Parallel Computing.* MIT Press, 1994.

[16] A. Grimshaw and W. Wulf. Legion – a view from 50,000 feet. In *Proc. 5th IEEE Symp. on High Performance Distributed Computing*, pages 89–99. IEEE Computer Society Press, 1996.

[17] W. Gropp, E. Lusk, and A. Skjellum. *Using MPI: Portable Parallel Programming with the Message Passing Interface.* The MIT Press, 1994.

[18] C. Lee, C. Kesselman, and S. Schwab. Near-Realtime Satellite Image Processing: Metacomputing in CC++. *Computer Graphics and Applications*, 16(4):79–84, 1996.

[19] M. Litzkow, M. Livney, and M. Mutka. Condor - a hunter of idle workstations. In *Proc. 8th Intl Conf. on Distributed Computing Systems*, pages 104–111, 1988.

[20] D. F. McMullen, G. von Laszewski Randall Bramley, I. Foster, and E. Westbrook. An NGI Application Testbed Partnership between Indiana University and Argonne National Laboratory: A Grid-based Collaboratory for Real-time Data Acquisition, Reduction and Visualization for Macromolecular X-Ray Crystallography Using the LBL Advanced. preprint, Indiana University and Argonne National Laboratory, April 1999.

[21] J. Morris, M. Satyanarayanan, M. Conner, J. Howard, D. Rosenthal, and F. Smith. Andrew: A distributed personal computing environment. *Communications of the ACM*, 29(3):184–201, 1986.

[22] SC98 High Performance Computing Challenge Best of Show award.

[23] SC'98 High Performance Computing Challenge: Industrial Mold Filling Simulation Using an Internationally Distributed Software Component Architecture. Web site, ://www.extreme.indiana.edu/sc98/sc98.html.

[24] P. Stelling, I. Foster, C. Kesselman, C.Lee, and G. von Laszewski. A Fault Detection Service for Wide Area Distributed Computations. In *Proc. 7th IEEE Symp. on High Performance Distributed Computing*, Aug. 1998.

[25] G. von Laszewski. The MDS Gridforum Working Group. http://www-fp.mcs.anl.gov/~gregor/mds.

[26] G. von Laszewski. Reusable Components of Globus^J. In *Workshop of Desktop Access to Remote Resources.* JavaGrande Forum and Argonne National Laboratory, http://www-fp.mcs.anl.gov/~gregor/javagrande, Oct. 1998.

[27] G. von Laszewski, S. Fitzgerald, B. Didier, and K. Schuchardt. *Roadmap to the Object Specification for Grid Information Services.* Argonne National Laboratory, Argonne, IL 60439, unpublished draft edition, 1999.

[28] G. von Laszewski, I. Foster, G. K. Thiruvathukal, and B. Toonen. A Computational Framework for Telemedicine. *Journal of Future Generation Computer Systems*, 14:pp. 10–123, 1998.

[29] G. von Laszewski, M.-H. Su, J. A. Insley, I. Foster, J. Bresnahan, C. Kesselman, M. Thiebaux, M. L. Rivers, S. Wang, B. Tieman, and I. McNulty. Real-Time Analysis, Visualization, and Steering of Microtomography Experiments at Photon Sources. In *Ninth SIAM Conference on Parallel Processing for Scientific Computing*, March 1999.

[30] G. von Laszewski, M. Westbrook, I. Foster, E. Westbrook, and C. Barnes. Using computational Grid Capabilities to Enhance the Cability of an X-Ray Source for Structural Biology. *to be published*, 1999.

[31] International union of crystallography. Web site, http://www.iucr.ac.uk/iucr-top/welcome.html.

[32] The American Crystallographic Association WWW. Web site, http://nexus.hwi.buffalo.edu/ACA/.

[33] S. Young, G. Fan, D. Hessler, S. Lamont, T. Elvins, M. Hadida, G. Hanyzewski, J. W. Durkin, P. Hubbard, G. Kindalman, E. Wong, D. Greenberg, S. Karin, and M. H. Ellisman. Implementing a Collaboratory for Digital Microscopy. *International Journal of Supercomputer Applications and High Performance Computing*, 2/3(10):170–181, 1996.