July 2006, Vol.21, No.4, pp.513–520 J. Comput. Sci. & Technol.

Globus Toolkit Version 4: Software for Service-Oriented Systems

Ian Foster

Computation Institute, Argonne National Laboratory & University of Chicago, Argonne, IL 60439, U.S.A.

E-mail: foster@mcs.anl.gov

Revised May 15, 2006.

Abstract The Globus Toolkit (GT) has been developed since the late 1990s to support the development of service-oriented distributed computing applications and infrastructures. Core GT components address, within a common framework, fundamental issues relating to security, resource access, resource management, data movement, resource discovery, and so forth. These components enable a broader "Globus ecosystem" of tools and components that build on, or interoperate with, GT functionality to provide a wide range of useful application-level functions. These tools have in turn been used to develop a wide range of both "Grid" infrastructures and distributed applications. I summarize here the principal characteristics of the recent Web Services-based GT4 release, which provides significant improvements over previous releases in terms of robustness, performance, usability, documentation, standards compliance, and functionality. I also introduce the new "dev.globus" community development process, which allows a larger community to contribute to the development of Globus software.

Keywords distributed systems, distributed applications, Internet applications, middleware, open source

1 Introduction

Globus is:

• A *community* of users and developers who collaborate on the use and development of open source software, and associated documentation, for distributed computing, virtual organizations, resource federation.

• The *software* itself—the **Globus Toolkit**: a set of libraries and programs that address common problems that occur when building distributed system services and applications.

• The *infrastructure* that supports this community—code repositories, email lists, problem tracking system, and so forth: all accessible at **dev.globus.org**.

The **software** provides a variety of components and capabilities, including:

• A set of *service implementations* that address resource management, data movement, service discovery, and related concerns.

• Tools for building *new Web Services*, in Java, C, and Python.

• A powerful standards-based *security infrastructure*, for authentication and authorization.

• Both *client APIs* (in different languages) and *command line programs* for accessing these various services and capabilities.

• Detailed *documentation* on these various components, their interfaces, and how they can be used to build applications.

These components in turn enable a rich **ecosystem** of components and tools that build on, or interoperate with, GT components—and a wide variety of **applications** in many domains. From our experiences and the experiences of others in developing and using these tools and applications, we identify commonly used design patterns or **solutions**, knowledge of which can facilitate the construction of new applications.

In the remainder of this article, I review briefly the current status of the Globus community, software, and infrastructure, focusing in particular on those aspects of GT4 that should be of interest to those wishing to work with the software. I reference more technical articles for more details on the underlying concepts and mechanisms.

2 Motivation and Concepts

Globus software is designed to enable applications that federate distributed resources, whether computers, storage, data, services, networks, or sensors. Initially, work on Globus was motivated by the demands of "virtual organizations"[1] in science. More recently, commercial applications have also become important. In both commerce and science, a need to accelerate the pace of innovation demands technologies that can reduce barriers to resource access and federation.

Federation is typically motivated by a need to access resources or services that cannot easily be replicated locally. For example:

• a scientist (or business analyst) needs to access data located in different storage systems across a scientific collaboration (or enterprise);

• a business (or physics community) must allocate computing, storage, and network resources dynamically for a time-varying e-commerce (or physics data analysis) workload;

Regular Paper

Work on Globus has been supported in part by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, U.S. Department of Energy, under Contract W-31-109-Eng-38, by the National Science Foundation (NSF)'s Office of Cyberinfrastructure and other programs, and by IBM, DARPA, NASA, Microsoft, the UK Engineering and Physical Sciences Research Council and Department of Trade and Industry, and the Swedish Research Council.

• an engineer needs to design and operate experiments on remote equipment, linking and comparing numerical and physical simulations;

• an astronomy experiment must replicate a terabyte of data a day to partner sites worldwide.

While every application has unique requirements, a small set of functions frequently recur. For example, we must often discover available resources, configure a computing resource to run an application or deploy a service, manage an application or service, move data reliably from one site to another, monitor system components, control who can do what, manage user credentials and attributes. Good-quality implementations of these functions can reduce development costs. Furthermore, if these implementations are widely adopted and/or implement standards, they can enhance interoperability. Globus software addresses both goals, using an open source model to encourage both contributions and adoption, and implementing standards whenever feasible.

GT4 makes extensive use of Web Services[2] to define its interfaces and structure its components. Web Services provide flexible, extensible, and widely adopted XML-based mechanisms for describing, discovering, and invoking network services. Its document-oriented protocols are well suited to the loosely coupled interactions that many argue are preferable for robust distributed systems[3]. These mechanisms facilitate the development of service-oriented architectures—systems and applications structured as communicating services, in which service interfaces are described, operations invoked, access secured, etc., in uniform ways.

While end-user *applications* are typically concerned with domain-specific operations such as pricing a portfolio or analyzing a gene sequence, computing ultimately requires the manipulation and management of *infrastructure*: physical devices such as computers, storage systems, and instrumentation[4]. Thus, GT4 provides a set of *infrastructure services* that implement interfaces for managing computational, storage, and other resources. In many Globus deployments, such as TeraGrid[5], Open Science Grid[6,7], Cancer Bioinformatics Grid (caBIG)[8], EGEE[9], LHC Computing Grid[10], UK National Grid Service[11], China Grid[12], China National Grid[13], and NAREGI[14], these services are deployed to support a range of different application communities, each of which then executes their own application-specific code that relies on those services.

3 Globus Architecture

Fig.1 illustrates various aspects of GT4 architecture. It depicts three sets of components as follows.

• A set of **service implementations** (the bottom half of the figure) implement useful infrastructure services. These services address such concerns as execution management (GRAM), data access and movement (GridFTP, RFT, OGSA-DAI), replica management (RLS, DRS), monitoring and discovery (Index, Trigger, WebMDS), credential management (MyProxy, Delegation, SimpleCA), and instrument management (GTCP). Most are Java Web Services but some (bottom right) are implemented in other languages and/or use other protocols.

• Three **containers** can be used to host user-developed services written in Java, Python, and C, respectively. These containers provide implementations of security, management, discovery, state management, and other mechanisms frequently required when building services. These containers extend open source service hosting environments with support for useful Web Service (WS) specifications, including WS Resource Framework (WSRF), WS-Notification, and WS-Security[15,16].

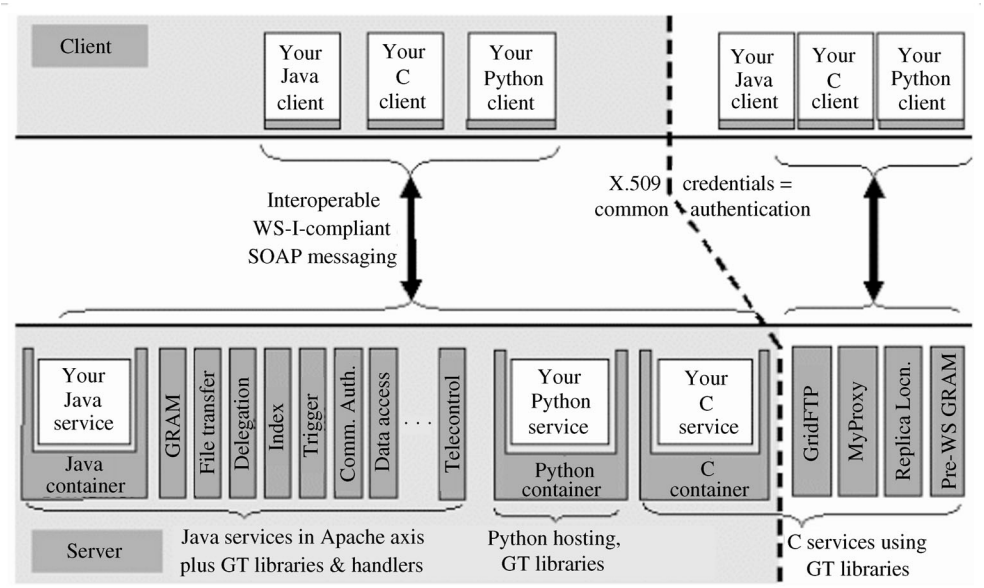


Fig.1. Selected GT4 components and interactions. Shaded boxes are GT4 code and white boxes user code.

• A set of **client libraries** allow client programs in Java, C, and Python to invoke operations on both GT4 and user-developed services. In many cases, multiple interfaces provide different levels of control: for example, in the case of GridFTP, there is not only a simple command-line client (globus-url-copy) but also control and data channel libraries for use in programs—and the XIO library allowing for the integration of alternative transports.

While individual GT4 services are useful in and of themselves, the whole is more than the sum of the arts. Uniform abstractions and mechanisms mean that clients can interact with different services in similar ways. This uniformity, which facilitates the construction of complex, interoperable systems and encourages code reuse, occurs at several levels:

• WS-I-compliant *SOAP messaging* among Web Services and their clients;

• a common *security and messaging infrastructure* enables interoperability among different applications and services;

• a powerful and extensible *authorization framework*[17] provides uniform, standards-based access to a range of different authorization mechanisms;

• all containers and most services implement common Web Services interfaces and behaviors for state representation, access, and subscription, facilitating *discovery and monitoring*[18];

• common abstractions and interfaces for *lifetime management* of stateful components, supporting both explicit and soft state destruction.

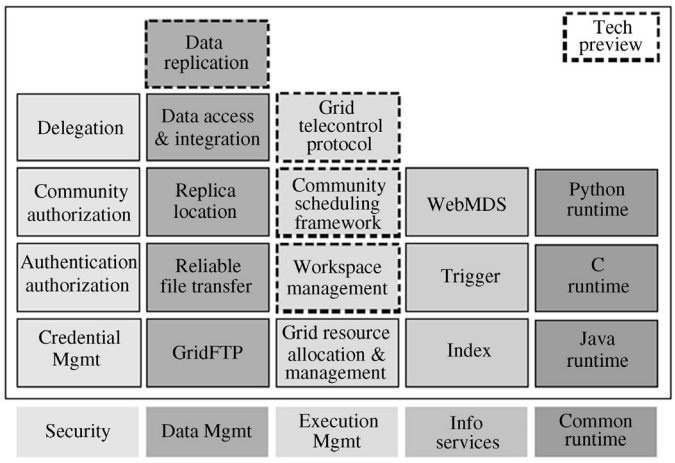


Fig.2. Primary GT4 components (dashed lines represent "tech previews").

4 Globus Software Details: How Do I ...?

Fig.2 provides another perspective on GT4 structure, showing the major components provided for basic runtime (on the right) and then (from left to right) security, execution management, data management, and information services. I introduce these components by showing how they can be used to perform various tasks.

4.1 How Do I Manage Execution?

Let us say we want to run a task on a computer, or deploy and manage a service that provides some capability to a community. In both cases, we need to acquire access to a computer, configure that computer to meet our needs, stage an executable, initiate execution of a program, and monitor and manage the resulting computation.

The GT4 Grid Resource Allocation and Management (GRAM) service[19] addresses these issues, providing a Web Services interface for initiating, monitoring, and managing the execution of arbitrary computations on remote computers. This interface allows a client to express such things as the type and quantity of resources desired, data to be staged to and from the execution site, the executable and its arguments, credentials to be used, and job persistence requirements. Other operations enable clients to monitor the status of both the computational resource and individual tasks, to subscribe to notifications concerning their status, and control a task's execution.

A GRAM service can be used for many different purposes. The following are some examples.

• The GriPhyN Virtual Data System (VDS)[20,21], Ninf-G[22], and Nimrod-G[23] use GRAM interfaces to dispatch (potentially large numbers of) individual tasks to computers. For example, GADU[24] uses VDS to dispatch several million BLAST and BLOCKS runs as it updates its proteomics knowledge base.

• GRAM is often used as a service deployment and management service. GRAM is used first to start the service and then to control its resource consumption and provide for restart in the event of resource or service failure.

• The MPICH-G2 implementation[25] of the Message Passing Interface uses GRAM to coschedule subtasks across multiple computers. For example, Dong *et al.*[26] have used MPICH-G2 to conduct a complete simulation of the human arterial tree.

Two additional components are provided within GT4 as "tech previews", meaning that they are less thoroughly tested than other components and more likely to change in the future:

• a Workspace Management Service (WMS)[27] provides for the dynamic creation of execution sandboxes, using virtual machines (e.g., Xen[28]) or Unix accounts;

• the Grid TeleControl Protocol (GTCP) service[29] is for managing instrumentation; it has been used for earthquake engineering facilities and microscopes.

All of these "execution management" services have in common that they create a "managed computation" within a specified execution environment, that can then be monitored and managed via Web Services interfaces.

4.2 How Do I Access and Move Data?

Globus applications often need to manage, provide access to, and/or integrate large quantities of data at one or many sites. This "data problem" is broad and complex, and no single piece of software can "solve" it in any comprehensive sense. However, several GT4 components implement useful mechanisms that can be used individually and in conjunction with other components to develop interesting solutions. (A recent article[30] reports on these tools and various success stories.)

• The Globus implementation[31] of the **GridFTP** specification provides libraries and tools for reliable, secure, high-performance memory-to-memory and disk-to-disk data movement. It has achieved 27 Gbit/s end-to-end over wide area networks, and can interoperate with conventional FTP clients and servers. GridFTP provides the substrate on which are built many higher-level tools and applications.

• The Reliable File Transfer (RFT) service[32] provides for the reliable management of multiple GridFTP transfers. It has been used, for example, to orchestrate the transfer of one million files between two astronomy archives.

• The Replica Location Service (RLS)[33] is a decentralized system for maintaining and providing access to information about the location of replicated files and datasets. For example, the LIGO experiment uses it to manage more than 40 million file replicas across 10 sites[34].

• The Data Replication Service (DRS: a tech preview) combines RLS and GridFTP to provide for the management of data replication[34].

• Data Access and Integration (OGSA-DAI) tools[35] provide access to, and server-side processing of, relational and XML data.

4.3 How Do I Monitor and Discover Services?

Monitoring and discovery are two vital functions in any distributed system, particularly when that system spans multiple locations, as in that context no-one is likely to have detailed knowledge of all components. Monitoring allows us to detect and diagnose the many problems that can arise in such contexts, while discovery allows us to identify resources or services with desired properties. Both tasks require the ability to collect information from multiple, perhaps distributed, information sources.

In recognition of the importance of these functions, monitoring and discovery mechanisms are built into GT4 at a fundamental level, as follows (see Fig.3)[36].

• GT4 provides standardized mechanisms for associating XML-based **resource properties** with network entities and for accessing those properties via either pull (query) or push (subscription). These mechanisms—implementations of the WSRF and WS-Notification specifications[15]—are built into every GT4 service and container, and can also be incorporated easily into any user-developed service. Services can be configured to register with their container, and containers with other containers, thus enabling the creation of hierarchical (or other) structures.

• GT4 provides two **aggregator services** that collect recent state information from registered information sources. As not all information sources support WSRF/WS-notification interfaces, these aggregators can be configured to collect data from any information source, whether XML-based or otherwise. The two aggregators implement a registry (**Index**) and event-driven data filter (**Trigger**), respectively.

• GT4 provides a range of browser-based interfaces, command line tools, and Web Service interfaces that allow users to query and access the collected information. In particular, the **WebMDS** service can be configured via XSLT transformations to create specialized views of Index data.

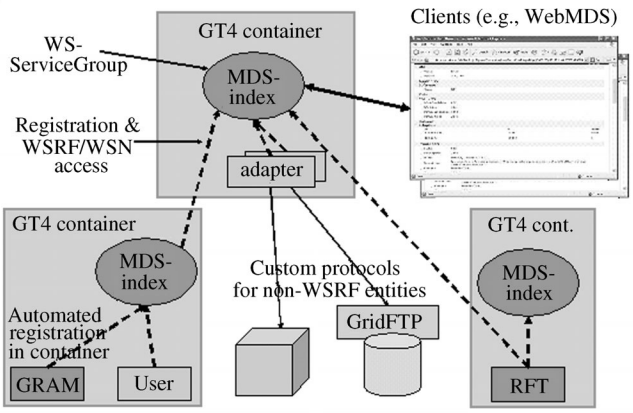


Fig.3. GT4 monitoring and discovery.

These mechanisms provide a powerful framework for monitoring diverse collections of distributed components and for obtaining information about those components for purposes of discovery. For example, the Earth System Grid (ESG)[37] uses these mechanisms to monitor the status of the services that it uses to distribute and provide access to more than 100 TB of climate model data.

4.4 How Do I Control Who Can Do What?

Security concerns are particularly important and challenging when resources and/or users span multiple locations. A range of players may want to exert control over who can do what, including the owners of individual resources, the users who initiate computations, and the "virtual organizations" established to manage resource sharing. "Exerting control" may include variously enforcing policy and auditing behavior. When designing mechanisms to address these requirements, we must work not only to protect communications but also to limit the impact of compromises at end systems. A complete security "solution" must combine components concerned with establishing identity, applying policy,

tracking actions, etc., to meet specific security goals. GT4 and related tools provide powerful building blocks that can be used to construct a range of such systems.

At the lowest level, GT4's highly standards-based security components implement credential formats and protocols that address message protection, authentication, delegation, and authorization. As shown in Fig.4, support is provided for (a) WS-Security-compliant message-level security with X.509 credentials (slow) and (b) with usernames/passwords (insecure, but WS-I Base Security Profile compliant) and for (c) transport-level security with X.509 credentials (fast and thus the default).

In GT4's default configuration, each user and resource is assumed to have an X.509 public key credential. Protocols are implemented that allow two entities to validate each other's credentials, to use those credentials to establish a secure channel for purposes of message protection, and to create and transport delegated credentials that allow a remote component to act on a user's behalf for a limited period of time[38,39].

Authorization call outs associated with GT4 services can be used to determine whether specific requests should be allowed. In particular, the *authorization framework* component[17] allows chains of authorization modules with well-defined interfaces to be associated with various entities, e.g., services, in the container. This component also provides multiple different authorization module implementations, ranging from traditional Globus gridmap-based authorization to a module that uses the SAML protocol to query an external service for an authorization decision.

Supporting tools, such as MyProxy[40], GridShib[41], KX509[42], VOMS[43], and PERMIS[44], support the generation, storage, and retrieval of the credentials that GT4 uses for authentication, and address issues concerning group membership, authorization policy enforcement, and the like. These tools can be used to build systems that provide secure authentication while never requiring users to manage their own X.509 credentials. In addition, they can integrate with systems such as Kerberos and Shibboleth.

4.5 How Do I Build New Services?

The GT4 distribution builds on (and includes) open source Web Services "container" software that supports the development of components that implement Web Services interfaces. This software deals with such issues as message handling and resource management, thus allowing the developer to focus their attention on application logic.

GT4 also packages additional components to provide *GT4 Web Services containers* for deploying and managing services written in Java, C, and Python. As illustrated in Fig.5, these containers can host a variety of different services as follows.

- Implementations of **basic WS specifications** such as *WSDL, SOAP, and WS-Security* support services that make use of these specifications to implement basic Web Services functionality.
- Implementations of **state management specifications**, notably *WS-Addressing, WSRF, and WS-Notification*, support services that want to expose and manage state associated with services, back-end resources, or application activities[15]. (For example, GT4 GRAM and RFT services use these mechanisms to manage state associated with tens of thousands of computational activities and file transfers, respectively.)
- The Java container hosts the **GT4 Java Web Services** mentioned earlier, such as GRAM, RFT, DRS, Delegation, Index, and Trigger.
- Enhanced **registry and management capabilities**, notably the representation of information about services running in a container as WS-Resources, facilitate the creation of distributed registries and system monitoring tools.

In general, the Java container provides the most advanced programming environment, the C container the highest performance (a detailed performed evaluation is provided by Humphrey *et al.*[45]), and (Python enthusiasts would argue) the Python container the nicest language. If developing new services in Java using GT4, see the tutorial text[46] and its accompanying Web site.

Numerous projects are developing exciting services and applications based on GT4 containers. For example, the Belfast eScience Center has 1.5 million lines of GT4 Java code (converted from GT3, a process that re-

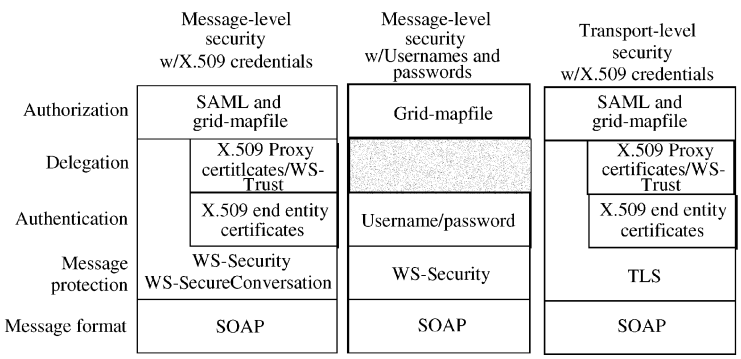


Fig.4. GT4 security protocols (see text for details). From [16].

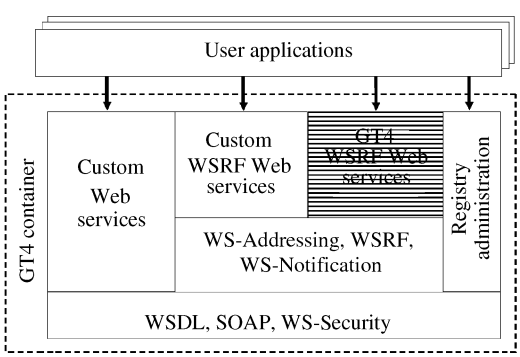


Fig.5. Capabilities of a GT4 container.

quired "relatively few changes in service code"[47]), implementing a range of applications including a digital video management system for the BBC, and the China Grid Support Package provides a rich set of services for eScience and education built on the GT4 Java container.

Several projects have developed interactive development environments (IDEs) for GT4 services. The Introduce system from Ohio State University is a mature example of such a system.

4.6 How Do I Do More Complicated Things?

GT4 services and libraries do not provide complete solutions to many distributed computing problems: to do anything more complex than submit a job or move a file, you must use GT4 software in conjunction with other tools and/or your own code—or access a (GT-based) service that provides the capabilities that you require[48].

In analyzing how people use Globus software, we find that similar patterns tend to reoccur across different projects and application domains. Thus, we have launched an effort to document these **solutions**[49] and how they can be implemented using components of the Globus ecosystem.

5 Processes, Results, and Evaluation

The Globus Alliance's **software engineering processes** have matured, driven by both increased resources and more aggressive users able to contribute to testing. These processes now include:

• extensive *unit test suites* and the use of *test coverage tools* to evaluate coverage;

• frequent *automated execution of build and test suites* on more than 20 platforms, via both local systems and the NMI GRIDS Center's distributed build/test facility;

• extensive *performance test suites* used to evaluate various aspects of component performance, including latency, throughput, scalability, and reliability;

• a cross-GT *documentation plan*, managed by a dedicated documentation specialist, to ensure complete coverage and uniform style for all components;

• a well-defined *community testing process*, which in the case of GT4 included a six-month alpha and beta-testing program with close to 200 participants;

• an *issue tracking system* based on bugzilla, used to track error reports and feature requests, and the work associated with those issues;

GT4 **performance** is discussed in various reports that address the performance of different Web Services containers, including GT4's Java, C, and Python[45]; the GT4 implementation of GridFTP[31]; and the GT4 replica location service[50].

The UK eScience program has released an **external evaluation** of GT4[47]. This detailed report speaks favorably of the overall quality, usability, and performance of the GT4 code and its documentation. It notes, for example, that "GT4 installation was straightforward", "GT4 services demonstrated significant improvements in performance and reliability over their GT3 versions", and "GT4 package descriptions were of a high quality, well structured, and accurate".

6 Contributing

A large and diverse Globus community is working hard to improve the scope and quality of the Globus software. I hope that you, the reader, will feel inspired to contribute also. There are several ways in which you can do so.

Use the software and report your experiences. Simply using the software and reporting back on your experiences, positive or negative, can be immensely helpful. Reports of problems encountered, particularly when well documented, help guide bug fixes and/or prioritize work on new features. Reports of successful deployments and applications can help justify continued support for the development of the software.

Develop documentation and examples. Despite considerable progress, we remain in desperate need of code examples and associated documentation that can help other users to start work with Globus software or related tools. Take the time to document your successful application, and you will be repaid in gratitude from other users.

Contribute to software development. The list of new features wanted by users is always far greater than current Globus developers can handle. You can contribute bug fixes, test cases, new modules, or even entirely new components. In early 2006, we unveiled "dev.globus" to facilitate such contributions: see the next section.

7 Dev.Globus Community

Dev.globus (see http://dev.globus.org), unveiled in early 2006, comprises both a governance process and an infrastructure designed to allow for broad contributions to Globus software. Modeled closely on Apache, dev.globus includes the following components.

• A modular architecture by which the Globus software is partitioned into a number of distinct projects, each with its own separately managed code base. (Within this structure, the Globus Toolkit exists as a project focused on creating and distributing a quality-controlled distribution that incorporates software from many, but not necessarily all, Globus projects.)

• A governance structure that places the technical decisions concerning each component in the hands of its principal contributors (its "committers"). A Globus Management Committee (GMC) provides architectural guidance and conflict resolution.

• An infrastructure for hosting Globus projects. This infrastructure includes source code repositories, email lists, wikis, and build and test mechanisms.

While dev.globus is (at the time of writing) still in its infancy, it is so far proving highly successful, with the number of both components and developers growing rapidly.

8 Related Work

Globus builds on a rich tradition of prior work in distributed systems. The metacomputing[51], Legion[52], and I-WAY[53] initiatives of the past decade have also been influential, as has contemporary work in such areas as Web Services, virtual machines, and peer-to-peer architecture. The PlanetLab community[54] is addressing similar concerns, as discussed in a recent article[55].

9 Futures

We are entering an exciting time for Globus, due to the confluence of the following factors:

- the completion of GT4 means that the Globus software now has a solid Web Services base on which to build new services and capabilities;
- sustained funding for eScience support will allow us to accelerate efforts aimed at meeting demands for ever-greater scalability, functionality, usability, and so forth;
- the creation of entities dedicated to the support needs of industry means that commercial adoption (and contributions) will accelerate;
- a rapidly growing user community is increasing the quantity and quality of user feedback, code contributions, and components within the larger Globus ecosystem;
- revisions to the Globus infrastructure and governance processes, via dev.globus, are allowing the set of contributors to the software and documentation to expand.

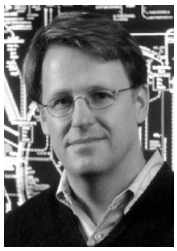
The Globus community is engaged in many activities aimed at developing the capabilities, usability, and range of application of the software. Examples include higher-level data management services, dynamic service deployment and management[56], interactive development environments, attribute and trust management for virtual organizations, more powerful and scalable workflow management, advance reservation and agreement negotiation for different resource types, and monitoring and problem determination.

Acknowledgements I report here on the work of many talented colleagues, as detailed at www.globus.org. The core team is currently based primarily at Argonne National Lab, U. Chicago, the USC Information Sciences Institute, U. Edinburgh, the Royal Institute of Technology, the National Center for Supercomputing Applications, and Univa Corporation, but many others have also contributed to Globus code, documentation, and testing, and/or made our work worthwhile by using the software.

References

[1] Foster I, Kesselman C, Tuecke S. The anatomy of the Grid: Enabling scalable virtual organizations. *International Journal of Supercomputer Applications*, 2001, 15(3): 200–222.
[2] Booth D, Haas H, McCabe F *et al.* Web Services Architecture. W3C, Working Draft, 2003. http://www.w3.org/TR/2003/WD-ws-arch-20030808/.
[3] Kendall S C, Waldo J, Wollrath A, Wyant G. A Note on Distributed Computing. Technical Report TR-94-29, Sun Microsystems, 1994.
[4] Foster I, Tuecke S. Describing the elephant: The different faces of IT as service. *ACM Queue*, 2005, 3(6): 26–29.
[5] The TeraGrid Project. 2006, www.teragrid.org.
[6] Open Science Grid (OSG). 2006, www.opensciencegrid.org.
[7] Foster I *et al.* The Grid2003 production Grid: Principles and practice. In *IEEE Int. Symp. High Performance Distributed Computing*, 2004, IEEE Computer Science Press.
[8] Cancer Bioinformatics Grid (caBIG). 2006, http://cabig.nci.nih.gov.
[9] Enabling Grids for eScience (EGEE). 2006, http://public.eu-egee.org.
[10] LHC Computing Grid. 2006, http://lcg.web.cern.ch/LCG.
[11] UK National Grid Service. 2006, http://www.ngs.ac.uk.
[12] China Grid Project. 2006, http://www.chinagrid.org.
[13] China National Grid. 2005, http://www.cngrid.org.
[14] NAREGI: National Research Grid Initiative. 2006, http://www.naregi.org.
[15] Foster I, Czajkowski K, Ferguson D *et al.* Modeling and managing state in distributed systems: The role of OGSI and WSRF. In *Proc. the IEEE*, 2005, 93(3): 604–612.
[16] Welch V. Globus Toolkit Version 4 Grid Security Infrastructure: A Standards Perspective. 2004, http://www.globus.org/toolkit/docs/4.0/security/GT4-GSI-Overview.pdf.
[17] Lang B, Foster I, Siebenlist F *et al.* A multipolicy authorization framework for grid security. In *The 5th IEEE Int. Symp. Network Computing and Applications*, 2006.
[18] Czajkowski K, Fitzgerald S, Foster I, Kesselman C. Grid information services for distributed resource sharing. In *The 10th IEEE Int. Symp. High Performance Distributed Computing*, 2001, IEEE Computer Society Press, LA, CA, USA, 2001, pp.181–184.
[19] Czajkowski K, Foster I, Kesselman C. Agreement-based resource management. In *Proc. The IEEE*, 2005, 93(3): 631–643.
[20] Deelman E, Singh G, Su M H *et al.* Pegasus: A framework for mapping complex scientific workflows onto distributed systems. *Scientific Programming*, 2005, 13(3): 219–237.
[21] Foster I, Voeckler J, Wilde M, Zhao Y. The virtual data grid: A new model and architecture for data-intensive collaboration. In *Conf. Innovative Data Systems Research*, CA, USA, 2003.
[22] Tanaka Y, Nakada H, Sekiguchi S *et al.* Ninf-G: A reference implementation of RPC based programming middleware for grid computing. *Journal of Grid Computing*, 2002, 1(1): 41–51.
[23] Abramson D, Giddy J, Kotler L. High performance parametric modeling with Nimrod/G: Killer application for the global grid? In *Proc. the Int. Parallel and Distributed Processing Symposium* (*IPDPS*), Cancun, Mexico, 2000, pp.520–528.
[24] Rodriguez A, Sulakhe D, Marland E *et al.* A grid-enabled service for high-throughput genome analysis. In *Workshop on Case Studies on Grid Applications*, Berlin, Germany, 2004.
[25] Karonis N, Toonen B, Foster I. MPICH-G2: A grid-enabled implementation of the message passing interface. *Journal of Parallel and Distributed Computing*, 2003, 63(5): 551–563.
[26] Dong S, G K, Karonis N. Cross-site computations on the TeraGrid. *Computing in Science & Engineering*, 2005, 7(5): 14–23.
[27] Keahey K, Foster I, Freeman T, Zhang X. Virtual workspaces: Achieving quality of service and quality of life in the grid. *Scientific Programming*, 2005, 13(4): 265–2756.

[28] Barham P, Dragovic B, Fraser K *et al.* Xen and the art of virtualization. *ACM Symposium on Operating Systems Principles*, Bolton Landing, NY, USA, 2003, pp.164–177.
[29] Pearlman L, Kesselman C, Gullapalli S *et al.* Distributed hybrid earthquake engineering experiments: Experiences with a ground-shaking grid application. In *The 13th IEEE Int. Symp. High Performance Distributed Computing*, Honolulu, Hawaii, 2004, pp.14–23.
[30] Allcock W, Chervenak A, Foster I *et al.* Data grid tools: Enabling science on big distributed data. *SciDAC Conference*, San Francisco CA, USA, 2005, Institute of Physics Conf. Series, 16: 571–575.
[31] Allcock B, Bresnahan J, Kettimuthu R *et al.* The Globus Striped GridFTP Framework and Server. In *Proc. the ACM/IEEE SC2005 Conf. High Performance Networking and Computing*, Seattle, USA, Nov. 2005, p.54.
[32] Allcock W, Foster I, Madduri R. Reliable data transport: A critical service for the grid. In *Building Service Based Grids Workshop*, 2004, Global Grid Forum 11.
[33] Chervenak A, Deelman E, Foster I *et al.* Giggle: A framework for constructing scalable replica location services. *SC'02: High Performance Networking and Computing*, Baltimore, Maryland, USA, 2002.
[34] Chervenak A, Schuler R, Kesselman C *et al.* Wide area data replication for scientific collaborations. In *The 6th IEEE/ACM Int. Workshop on Grid Computing*, 2005.
[35] Atkinson M, Chervenak A, Kunszt P *et al.* Data Access, Integration, and Management. The Grid: Blueprint for a New Computing Infrastructure, Morgan Kaufmann, 2004.
[36] Schopf J M, Raicu I, Pearlman L *et al.* Monitoring and Discovery in a Web Services Framework: Functionality and Performance of Globus Toolkit MDS4. Technical Report, Mathematics and COmputer Science Division, Argonne National Laboratory, 2006.
[37] Bernholdt D, Bharathi S, Brown D *et al.* The earth system grid: Supporting the next generation of climate modeling research. In *Proc. the IEEE*, 2005, 93(3): 485–495.
[38] Gasser M, McDermott E. An architecture for practical delegation in a distributed system. In *1990 IEEE Symp. Research in Security and Privacy*, IEEE Press, 1990, pp.20–30.
[39] Foster I, Kesselman C, Tsudik G, Tuecke S. A security architecture for computational grids. In *The 5th ACM Conf. Computer and Communications Security*, 1998, pp.83–91.
[40] Novotny J, Tuecke S, Welch V. An online credential repository for the grid: MyProxy. In *The 10th IEEE Int. Symp. High Performance Distributed Computing*, San Francisco, 2001, IEEE Computer Society Press.
[41] Welch V, Barton T, Keahey K *et al.* Attributes, anonymity, and access: Shibboleth and Globus integration to facilitate grid collaboration. *PKI Conference*, 2005.
[42] NSF Middleware Initiative. KX.509/KCA, 2002, http://www.nsf-middleware.org/documentation/KX509KCA/.
[43] EU DataGrid VOMS Architecture v1.1. 2003, http://grid-auth.infn.it/docs/VOMS-v1_1.pdf.
[44] Chadwick D W, Otenko A. The PERMIS X.509 role based privilege management infrastructure. In *The 7th ACM Symposium on Access Control Models and Technologies*, Monterey, USA, 2002, pp.135–170.
[45] Humphrey M, Wasson G, Jackson K *et al.* A comparison of WSRF and WS-notification implementations: Globus toolkit V4, pyGridWare, WSRF: Lite, and WSRF. NET. In *The 14th IEEE Int. Symp. High Performance Distributed Computing*, Research triangle Park, NC, USA, 2005.
[46] Sotomayor B, Childers L. Globus Toolkit 4: Programming Java Services. Morgan Kaufmann, 2005.
[47] Harmer T, Stell A, McBride D. UK Engineering Task Force Globus Toolkit Version 4 Middleware Evaluation. UK Technical Report UKeS_2005-03, 2005.
[48] Foster I. Service-oriented science. *Science*, 2005, 308: 814–817.
[49] Grid Solutions. 2005, http://www.globus.org/solutions.
[50] Chervenak A L, Palavalli N, Bharathi S *et al.* Performance and scalability of a replica location service. In *The 14th IEEE Int. Symp. High Performance Distributed Computing*, Honolulu, Hawaii, 2004.
[51] Catlett C, Smarr L. Metacomputing. *Communications of the ACM*, 1992, 35(6): 44–52.
[52] Grimshaw A S, Wulf W A. The legion vision of a worldwide virtual computer. *Communications of the ACM*, 1997, 40(1): 39–45.
[53] DeFanti T, Foster I, Papka M *et al.* Overview of the I-WAY: Wide area visual supercomputing. *International Journal of Supercomputer Applications*, 1996, 10(2): 123–130.
[54] Bavier A, Bowman M, Chun B *et al.* Operating system support for planetary-scale services. In *1st Symposium on Network Systems Design and Implementation*, 2004, pp.253–266.
[55] Ripeanu M, Bowman M, Chase J *et al.* Comparing globus and PlanetLab resource management solutions. In *The 13th IEEE Int. Symp. High Performance Distributed Computing*, Honolulu, Hawaii, 2004, pp.246–255.
[56] Qi L, Jin H, Foster I, Gawor J. HAND: Highly Available Dynamic Deployment Infrastructure for Globus Toolkit 4. 2006.



Ian Foster received a B.S. (Hons I) degree in computer science from the University of Canterbury in Christchurch, New Zealand and a Ph.D. degree in computer science from Imperial College, London. Foster is a senior scientist at Argonne National Laboratory, Arthur Holly Compton distinguished service professor of computer science at the University of Chicago, and director of the Computation Institute at the University and Argonne. Foster's research interests are in distributed and parallel computing and computational science. He has published six books and over 300 articles and technical reports on these and related topics. Foster is also chair of the Globus Management Committee that leads the Globus community, and is Chief Open Source Strategist at Univa Corporation, a company he co-founded to foster and promote commercial applications of Grid technology. Dr. Foster is a fellow of the American Association for the Advancement of Science and the British Computer Society. His awards include the British Computer Society's award for technical innovation, the Global Information Infrastructure (GII) Next Generation award, the British Computer Society's Lovelace Medal, R&D Magazine's Innovator of the Year, and DSc Honoris Causa from the University of Canterbury.