Globus and PlanetLab Resource Management Solutions Compared

Matei Ripeanu
The University of Chicago
matei@cs.uchicago.edu

Mic Bowman
Intel Research
mic.bowman@intel.com

Jeffrey S. Chase
Duke University
chase@cs.duke.edu

Ian Foster
The University of Chicago & Argonne National Lab.
foster@cs.uchicago.edu

Milan Milenkovic
Intel Research
milan.milenkovic@intel.com

Abstract

PlanetLab and Globus Toolkit are gaining widespread adoption in their respective communities. Although designed to solve different problems– PlanetLab is deploying a worldwide infrastructure testbed for experimenting with network services, while Globus is offering general, standards-based, software for running distributed applications over aggregated, shared resources—both build infrastructures that enable federated, extensible, and secure resource sharing across trust domains. Thus, it is instructive to compare their resource management solutions. To this end, we review the approaches taken in the two systems, attempt to trace back to starting assumptions the differences in these approaches, and explore scenarios where the two platforms can cooperate to the benefit of both user communities. We believe that this is a key first step to identifying pieces that could be shared by the two communities, pieces that are complementary, and how Globus and PlanetLab might ultimately evolve together.

1. Introduction

The PlanetLab project is deploying and managing a worldwide infrastructure testbed for experimenting with a new class of network services. The Globus Alliance is developing a general, standards-based, software toolkit for running distributed applications over aggregated, shared resources. The two systems have many similarities in their user communities, goals, approaches, and technologies, but also important differences.

In this paper, we take a first step towards elucidating these commonalities and differences by undertaking a comparison of the approaches to resource management in the two systems. Although resource management is neither the complete nor final goal of either project, from a resource management perspective both PlanetLab and Globus attack similar problems: both need to discover, monitor, and allocate resources to applications/services in a coordinated, secure, and resilient fashion. It is therefore natural to compare the two systems to understand differences in the underlying goals, premises, and assumptions, and how these technical differences shape the two evolving architectures. Indeed, we believe that this understanding is key to identifying which pieces could transfer across domains (e.g., which wheels might one community reinvent, or avoid reinventing), which pieces are complementary, and how Globus and PlanetLab might ultimately evolve together.

Before proceeding with this comparison, we note three caveats. First, both Globus and PlanetLab are active research projects. Thus, we attempt to compare *both* their *existing* and their *planned* functionality and features. Moreover, aspects of this comparison are likely to become obsolete as the two projects evolve.

Second, while we focus here on comparing and contrasting resource management abstractions and mechanisms, the two projects are to a large degree complementary: Globus and Open Grid Services Architecture (OGSA) define protocols, interfaces, and behaviors for distributed resource management (e.g., WS-Agreement [5]) from which distributed systems can be constructed. PlanetLab developers, on the other hand, focus to a larger degree on implementing interfaces/behaviors to manage local systems with global behaviors left to the services built above this common base.

Table 1: Abbreviations used.

GT	Globus Toolkit [1]
GT3	Globus Toolkit version 3 [1]
VO	Virtual Organization
WSRF	Web Services Resource Framework
OGSA	Open Grid Services Architecture [3]
GSI	Grid Security Infrastructure [4]
VM	Virtual Machine

Third, key differences ultimately influence the two solutions: *Globus is a software toolkit that is based on standards and has deployments. PlanetLab is a deployment that has a software system and may ultimately influence or produce standards.* For example, GT3 has *multiple* deployments while PlanetLab, at least in its current instantiation, is building the equivalent of a *single* deployment. The PlanetLab Consortium produces the PlanetLab software and manages its single deployment on a rather homogeneous hardware/software base. In contrast, the multiple deployment assumption requires Globus developers to work with fewer assumptions on participating resources, on existing infrastructure deployments (e.g., security infrastructure), or on the performance parameters of these deployments should achieve.

The approach to standardization, perhaps a side effect of different maturity stages, has a similarly strong influence: the Globus project works closely with Global Grid Forum [6], OASIS, IETF, and W3C to define standards and gain community acceptance. PlanetLab infrastructure solutions are based on "rough consensus and working code" and focus on efficient testbed operations; they might ultimately influence or produce standards but PlanetLab considers the infrastructure to be an open research topic that would be hindered by early standardization.

With these caveats in mind, we now proceed to our comparison. We briefly describe Globus and PlanetLab (Section 2), contrast their starting assumptions (Section 3), decompose solutions into basic mechanisms that we compare, try to highlight what appears to be a valuable technique for a particular sub-domain (Section 4), and present a scenario where Globus and PlanetLab can work together to provide services that are more valuable than either in isolation (Section 5).

2. Background

We first provide some background information on the Globus and PlanetLab systems.

2.1. Grids and the Globus Toolkit

Grids aim to enable "resource sharing and coordinated problem solving in dynamic, multi-institutional virtual organizations" [7]. In other words, grids provide an infrastructure for federated resource sharing across trust domains. Grids evolved from the idea of *metacomputing* [1, 8]: building a uniform computing environment from diverse resources by defining standard network protocols and/or interposing a uniform API at the library level. Much like the Internet on which they build, current grids define protocols and middleware that can mediate access to a wide range of resources without requiring modifications to operating systems. Applications use services provided by this layer to discover, aggregate, and harness resources.

The recently proposed WS-Resource Framework (WSRF) and its implementation in the Globus Toolkit v4, among others, define uniform mechanisms for managing remote state, creating a standard substrate for building virtual organizations (VOs) and developing new services and applications that exploit the resources shared within these VOs.

WSRF and related Web Services and OGSA standards [3] are crucial to the Grid vision; they are the standards that make it possible to develop large-scale, reliable, and interoperable grid applications and services. However, these standards are largely independent of the underlying resource management mechanisms used. Thus the rest of this document will discuss them only superficially as we focus on *mechanisms* rather than standards or protocols.

Globus Toolkit [1] is a collection of technologies (in their most recent instantiation, Web services-based and WSRF-compliant) that provides basic middleware to create VOs, addressing such issues as security, resource discovery, resource management, and data movement. At deployment, depending on available resources and planned applications, specific service implementations can be chosen and deployed, often in conjunction with other GT-based components. GT is in production use across VOs integrating resources from 20-50 sites [9-13] with thousands of computational and data resources, and is expected to scale to 100s of sites with 1000s of sites as a future goal.

2.2. PlanetLab

PlanetLab [14, 15] is a large-scale, distributed platform for new network services such as content distribution networks [16-18], robust routing overlays [19], network measurement services [20-22], scalable object location [23-26], network embedded storage [27], and application-level multicast [28, 29]. PlanetLab was envisioned as a global testbed for developing and deploying next-generation Internet services and offering them to others for experimental use and eventually perhaps for production use. The current PlanetLab user community consists primarily of researchers in networking and distributed systems, although PlanetLab may host services with user communities who are unaware of its existence. The testbed is best suited to services that need multiple, possibly geographically dispersed "points of presence."

PlanetLab is designed to run on dedicated hosts. It provides purpose-built software from the ground-up, including an operating system (currently modified Linux) with extensions for virtualization. PlanetLab

uses virtualization containers to manage resource allocation and to achieve isolation between a potentially large number of long-lived, independent services.

PlanetLab provides its users with a virtual container at each host to act as a "point of presence" for a service. From a service programmer's perspective, PlanetLab provides a distributed virtual machine with a relatively low-level system abstraction, in the form of (a distributed set of) virtual containers and a familiar Unix-style API. It is envisaged that high-value services, such as storage or naming, will be built by the user community, and that successful ones will eventually be incorporated into the common core.

PlanetLab currently includes more than 370 hosts at over 155 sites and is planned to grow to about 1000 sites with a few nodes each plus a small number of sites with more substantial computing resources (e.g., clusters). A significant part of PlanetLab infrastructure is dedicated to managing resources both at the node level and in the aggregate.

3. Different starting assumptions …

While the Globus and PlanetLab efforts tackle similar resource management problems they make further assumptions regarding resources and application requirements that sometimes lead to different solutions. Their starting assumptions differ in a number of key areas: the user communities they serve, the characteristics of the most frequent applications and resources, and the degree of control individual sites retain over resources made available to a VO.

3.1. User communities

PlanetLab and Globus serve distinct, although overlapping, user communities. The PlanetLab user community comprises primarily computer science researchers interested in experimenting with *infrastructure* for building "planetary scale" services. The Globus user community is a heterogeneous pool of end-users (in science and industry), including computer scientists, interested in efficiently running their *end-user applications*. This distinction results in different functionality, as noted in the following.

- PlanetLab itself provides only minimal functionality, leaving services unconstrained in the way they provide richer functionality to applications. One downside, at least in the short term, may be duplicated user effort when the same functionality is implemented in multiple services. The potential upside is having multiple implementations of similar functionality emerging as competing *services* that are selected by application writers based on merit.
- Globus, on the other hand, aims to provide richer, standardized functionality closer to current application requirements – although there does also exist a rich ecology of higher-level tools and services that build on Globus mechanisms to address specific application requirements.

One example is the security infrastructure: PlanetLab provides limited security functionality and services build their own security layer if needed (e.g., the SHARP [2] resource trading framework develops its own trust delegation and authentication mechanisms in the PlanetLab context). In contrast, Globus Toolkit's Grid Security Infrastructure (GSI [4]) framework includes a complete machinery with protocols, APIs, and tools based on WS-Security mechanisms.

3.2. Application characteristics.

The applications and services targeted by the two communities have different characteristics that generally result in different resource requirements.

- Grid applications are often compute-intensive [10-12], although some also consume significant amounts of disk and/or network bandwidth as a result of focusing on, for example, integration of large-scale data repositories (data grids [30-33], virtual observatory [34]), collaboration [35], or the control of scientific instruments [36, 37].
- PlanetLab services are generally network-intensive and rarely have significant CPU demands. Experimental services include network measurement [20-22], application-level multicast [28, 29], DHTs [23-26], storage [27, 38-40], resource allocation [2], distributed query processing [41-43], content distribution networks [16-18], monitoring [44], and overlay networks [19, 45].

On another axis, the two communities take different approaches to geographical resource distribution and to resource partitioning among services/applications. Roughly the difference can be summarized as follows: for PlanetLab services, embracing resource distribution is an objective, while for grid applications, resource distribution is a necessary evil.

For some classes of PlanetLab services (e.g., network monitoring services) wide geographical distribution is essential. For grids, geographic resource distribution is typically a consequence of VO membership and rarely an application requirement. (There are exceptions, e.g., content distribution or collaboration applications.)

Given a choice, few current grid applications will prefer to operate over a large set of resources with limited capabilities. In contrast, most network services

envisioned for PlanetLab try to exploit the wide-area distribution (e.g., multiple network vantage points) and the (presumably) uncoordinated failures offered by a large set of resources, even if these resources come with limited individual capabilities.

3.3. Resources

PlanetLab's mission as a testbed and deployment platform for a *new* class of network services allows for little resource heterogeneity in the underlying infrastructure: no legacy hardware or software has to be supported. PlanetLab, assumes (and exploits) this lack of heterogeneity. For example, the security infrastructure is based on SSH, which excludes sites that require a different security model (say Kerberos) or certificate standard (e.g., X.509, PGP, SPKI).

Currently, PlanetLab supports Intel-based desktop and server configurations and one operating system (Linux).

Globus can operate on a wide range of devices: clusters, workstations, PDAs, file systems, databases, sensors, and scientific instruments can all be integrated into a VO. All major operating systems are supported in GT2 and the Java-based implementation of OGSA standards in GT3 and GT4 further expands the set of environments where Globus can be deployed.

3.4. Resource ownership

Both Globus and PlanetLab aim to allow participating sites to retain control over local resources, e.g., by allocating local resources with site-specific priorities, by black- or white-listing users at the site level, or by specifying and enforcing site-specific usage policies.

In the tradeoffs space (Figure 1) between autonomy offered to individual sites and the functionality that can be built at the federation level, Globus and PlanetLab make distinct decisions, as follows:

PlanetLab limits the control individual sites have over their own resources in a number of ways: by mandating the operating system and key components of the security infrastructure, by allowing PlanetLab administrators 'root' access on individual nodes, and by giving PlanetLab administrators access to a remote power button for each site. These decisions enable a faster evolution of the testbed, as a more compact set of software can evolve more quickly and software updates can be easily distributed and deployed by central administrators.

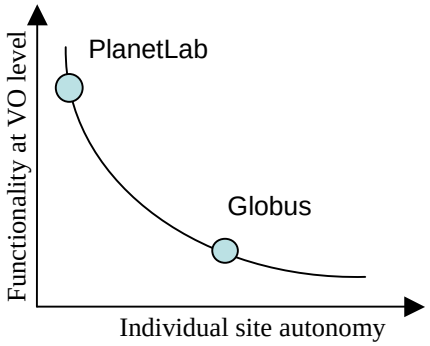

Figure 1: PlanetLab and Globus make different decisions when balancing between individual site autonomy and functionality offered at the VO level.

In contrast, when using Globus, individual sites that bring resources to a VO typically relinquish much less control to external organizations—they might permit application communities to use tools such as Pacman to automate the deployment of *application* software to grid resources, but privileged services are firmly under the control of local site administrators.

In this respect, one can say that PlanetLab emphasizes global coordination over local autonomy to a greater degree than Globus; PlanetLab sites relinquish more control to external administrators.

4. … lead to different solutions

Both PlanetLab and Globus are built using orthogonal sets of mechanisms that can be naturally grouped into two categories: mechanisms for managing resources at the individual node (or site) level and mechanisms that enable federated sharing of resources (i.e, for building virtual organizations). We compare Globus and PlanetLab solutions in each category.

4.1. Local resource management abstractions

The two platforms have different foci. GT focuses on integrating, to the extent possible, existing resources with their hardware, operating systems, and local resource management and security infrastructure. GT provides, in effect, a set of unifying interfaces through which local resource management functionality can be discovered and used. (Some GT communities require standardization in hardware and define standard software suites that may include local resource management functions, as in NEESgrid [46] and BIRN [47]. However, it is rare that there is not some amount of heterogeneity to manage.)

Assume, for example, an application that runs one hour starting at midnight every day for a week on the same node. The manager or user of such an application must discover a node that supports reservations, query for available timeslots, make a reservation, claim the reservation each day, and bind it to the application, with all of these functions accessed via standard protocols that map to node-specific functionality.

PlanetLab, in contrast, specifies the individual node architecture and functionality from the hardware level

up. The result is a platform where all participating nodes/sites provide uniform individual resource management functionality [48]. As a result, PlanetLab does not need to build the 'glue' level that GT provides to enable uniform access to, and management of, a heterogeneous set of resources.

The main abstraction offered by a PlanetLab node is a virtual machine (VM): each user of a PlanetLab node is presented with the image of a raw, dedicated machine. Currently the interface is the familiar Unix API, but in the future it will likely be a true virtual machine with improved isolation and better user control over the operating system and local resources. The emphasis here is on simplicity and generality on the assumption of a homogeneous hardware/software base: Intel-based servers running software whose bottom layer is dictated by PlanetLab.

In contrast, Globus evolved from metacomputing [8], the idea of building a uniform computing environment from diverse resources by interposing a uniform API at the library level and standard protocols at the network layer [49]. Thus Globus can embrace-and-extend the full range of deployed systems, including legacy OS and security architectures. The corresponding abstractions offered by the Globus Toolkit are the service (for GT3) or job (for GT2 and GT3). GT3 service interfaces are being defined not only for management of 'jobs' but also for managing computational resources, via for example the creation and initialization of a new virtual machine [50].

All local authorization and resource allocation decisions revolve around these abstractions: Is the user allowed to create a VM or invoke an operation on a grid service (run a job) on this node? Is a VM or grid service allowed to access certain resources? How are resource allocations specified and then bound to a VM or to a service/job? These questions are active research topics within the Globus and PlanetLab communities.

4.2. Global federation-building mechanisms

Delegation [51] is a key mechanism for enabling federated sharing of resources. In the rest of this section we compare the delegation approaches used in Globus and PlanetLab and show how they are exploited in the two systems to build global resource allocation/scheduling services.

4.2.1. Delegation mechanisms are essential to building federations. Resource management functionality at the VO level is generally based on (1) resource usage delegation: the ability of a node/site to delegate the right to consume its resources and/or (2) identity delegation: principals' ability to delegate their identity to other principals to act on their behalf.

Resource usage delegation

Both PlanetLab and Globus projects are developing mechanisms and protocols to enable a node or a site-wide resource manager to delegate resource consumption rights to an application or to a broker.

PlanetLab builds on *resource capabilities* [48] to offer the basic mechanism for resource usage delegation. PlanetLab resource capabilities represent time-limited claims over low-level resources available at a node or site: fair-share or dedicated use for CPU, network, memory, disk, network ports, file descriptors, etc. A local resource manager keeps track of resources available at a node and hands over capabilities to brokers that operate at the VO level.

A PlanetLab capability is represented by a 160-bit opaque identifier. Services that use and transfer these capabilities might add a more detailed description of the underlying resource together with authentication, authorization, or trust building mechanisms. (PlanetLab however does not standardize at this level.) SILK [52], a Linux kernel module, is the OS-level mechanism that supports and enforces capabilities.

At a higher-level, the corresponding solution under development in the grid community is the WS-Agreement protocol [5]. The goal of WS-Agreement is to define a uniform representation of agreements between resource/service providers and consumers and to formalize the negotiation process used to establish and modify agreements [53]. Note that a capability is in fact an implied agreement: the issuer of the capability agrees to provide some specified resources during a specified time interval to the capability holder.

WS-Agreement specifies a standard representation for agreements as Web Services, a (re)negotiation protocol, agreement states and their lifetimes, a standard way to describe agreement monitoring services, etc. The enforcement mechanism on the provider side is not specified: it can be a PlanetLab capability, a queuing system supporting reservations on a cluster, or any ad-hoc solution.

Note that these two efforts are complementary: PlanetLab focuses on implementing capabilities for various resource types and on integrating the fine-grained resource control they offer with VM functionality; WS-Agreement focuses on uniform agreement representation, naming and lifecycle.

Both PlanetLab and Globus intend to use capabilities or agreements to enable resource delegation. In PlanetLab a resource owner (the *node manager*) delegates the right to use a local resource by handing the corresponding capability to a user or service. Users/services acquire required capabilities directly from node managers or from specialized brokers that trade capabilities and then bind these

capabilities to a VM. The scenario imagined for WS-Agreement is similar: users negotiate agreements with resource owners and may later bind these agreements to submitted jobs or other running services.

Identity delegation

In many resource federation scenarios, a principal needs to perform certain actions on behalf of another principal. For example: user X calls service S1 and S1 needs to call S2 on behalf of X. If accounting or authorization decisions made at S2 depend on the original caller identity (X), then S1 needs to provide S2 an unforgeable, unrepudiable claim that the call is made on behalf of X. Most Globus-compatible schedulers are built using identity delegation: the scheduler receives jobs descriptions from users and submits them to individual sites on behalf of these users. This focus on identity delegation is motivated in part by the frequent requirement to be able to associate resource usage with specific individuals rather than communities or services. (The related Community Authorization Service [54] implements a capability-based service). We briefly summarize below the existing functionality.

The Grid Security Infrastructure (GSI) [4, 55, 56] uses time-limited proxy certificates [57], stored with unencrypted private keys, to address the identity delegation issue. These certificates are correctly formatted X.509 certificates [58], except that they are marked as proxy certificates and are signed by the user that delegates its identity rather than a Certificate Authority. Choosing the lifetime of proxy certificates requires a compromise between allowing long-term jobs to continue to run as authenticated entities and the need to limit the damage in the event a proxy is compromised. Proxy certificates with restricted rights are another way of limiting the potential damage caused by a stolen proxy. Authorization software run by relying parties recognizes proxy certificates and searches the certificate chain until the user certificate is found in order to do the authorization based on that identity token.

PlanetLab currently does not provide a mechanism for identity delegation. However, services can implement their own mechanisms.

4.2.2. Global resource allocation / scheduling. Strictly speaking, schedulers and resource allocation brokers are not part of either Globus or PlanetLab. However, it is relevant to compare existing and planned implementations for the two platforms. Together with resource management mechanisms at the individual node level, the ability to delegate is key to coordinated resource management at the VO level.

Generally schedulers built for PlanetLab employ resource usage delegation while those built for Globus exploit identity delegation. PlanetLab node managers and brokers push capabilities (resource reservations) from resources to the users that originate requests.

- *Existing functionality* is, in practice, primitive: most resources allocations are 'best-effort' and resources that cannot be shared (e.g., network ports) are allocated on a first-come-first-served basis.
- *Planned functionality*: multiple brokers/schedulers, presumably using different incentive models, implementing different allocation policies, or having application specific knowledge, will operate independently and dynamically share PlanetLab resources. This arrangement is enabled by the ability of each site/node to delegate resource usage rights to multiple brokers at fine granularity.
  This vision is quickly materializing: SHARP ([2], presented in more detail in Figure 2) is an example of a resource allocation framework currently developed for PlanetLab. With SHARP, sites can trade resources with dynamically discovered partners or contribute resources to federations according to local policies. Multiple resource management systems may coexist—either above or alongside SHARP—operating in independent PlanetLab slices. Application services such as batch schedulers may also exist within slices, scheduling the resources under their control according to local policies [59].

In Globus the flow is reversed: brokers pass job requests from users or applications to resources –

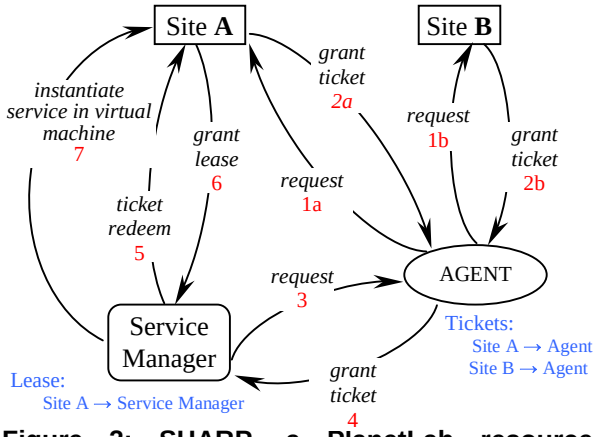

Figure 2: SHARP, a PlanetLab resource management example. A broker (agent) acquires tickets representing resources from sites A and B (steps 1 and 2). An application acquires these tickets from the broker (3, 4) and tries to redeem the tickets at their issuers for hard resource reservations (or leases) (5, 6). Once the application has obtained the leases it can create a VM, bind the resources represented by the leases to the VM (7), and start a service. Note that this mechanism allows individual sites/nodes to split their resources and distribute them to multiple brokers that operate independently. (Source: [2])

ideally based on knowledge of both resource availability and allocation policies. Most such brokers use identity delegation: a user sends a job description to the broker, which either submits the job or forwards it to another broker, all using the delegated identity of the user that originated the job. In some environments, the broker may forward the job under its own identity or under the identity of the VO with which the user is associated.

- *Existing functionality*: Numerous VO-level schedulers (or 'meta-schedulers'), some domain- or application-specific, have been developed by various groups (e.g., Nimrod-G [60], GrADS [61], DAGman and Condor-G [62], ASCI DRM [11], [63], EU DataGrid [31]). In addition, GT includes a co-allocation broker, DUROC [64].
- *Planned functionality*: as one example of efforts within the community, Platform Computing is developing the Community Scheduler Framework [65] based on the WS-Agreement specification.

5. PlanetLab and Globus together

The preceding discussion has focused on comparing and contrasting PlanetLab and Globus resource management solutions. However, we view the two efforts as compatible and complementary rather than competing. PlanetLab can be considered as an instance of a larger Grid agenda that attempts to simplify deployment for a narrower range of platforms and applications. It is focusing on developing capabilities (e.g., wide-area monitoring and instrumentation) that extend a piece of the Grid agenda, without necessarily excluding other pieces. As a platform, PlanetLab enables the layering of Globus—or of any alternative environment for interoperable heterogeneous computing—above the primitive support it provides. In other words, PlanetLab is just one of many platforms that can host Globus, and Globus is just one distributed systems environment that can run over PlanetLab. Indeed, we have installed Globus on the PlanetLab testbed and built the machinery to enable user access to this deployment as a service [50].

We present here one scenario in which Globus and PlanetLab complement each other to provide benefits that neither would be able to provide alone. Data grids [66] address a problem occurring in an increasing number of fields in which large data collections are important community resources. These data collections can be large (10^{12} to 10^{15} bytes) and are almost always geographically distributed, as are the computing and storage resources that scientific communities rely upon to store and analyze them.

Globus is being used to build VOs around these shared data collections and to implement services for distributed data management, access, and analysis. In addition to security mechanisms or the ability to define access policies at the VO level (supported by the Community Authorization Service [54]) GT includes tools aimed at high-performance data transfers (e.g., GridFTP a reliable file transfer service [66-68]). These tools are integrated with Globus security and authorization infrastructure and can split data transfers over multiple TCP streams to increase transfer throughput when data is striped across multiple nodes at both ends. We can imagine experimental PlanetLab services such as mTCP [69] and BANANAS [70] being used to optimize such transfers, by monitoring the Internet and using multipath routing to improve transfer throughput between two endpoints.

We believe that layering Globus on top of PlanetLab can significantly strengthen the data grid infrastructure. This architecture will benefit from the large PlanetLab deployed base and its ability to monitor the Internet to offer low-level networked services with improved performance and reliability, as in the case of the multi-path TCP data transfer service discussed above. Building on this set of lower level services, Globus can add a mature, widely adopted security infrastructure and higher-level services that are well integrated with user applications.

More generally, data grids face the need to develop and deploy distributed services of various sorts, such as resource discovery (e.g., Giggle [71]), data distribution (e.g., Kangaroo [72, 73], Stork [74]), and the data movement services mentioned above. PlanetLab can contribute here in two ways: as a community, it can be a source of ideas and expertise; as a deployment, it can potentially be a place to deploy these services "in the network" rather than at the edges of the network as is currently being done within projects such as Grid3 [75] or EU DataGrid [31].

On the other hand, PlanetLab could benefit from Globus experience in promoting service interoperability through uniform handling of identities and authentication, and of service discovery, representation, and invocation.

6. Summary and recommendations

We have reviewed the approaches taken to resource management in the Globus and PlanetLab systems, attempted to trace back to starting assumptions the differences in these solutions, and explored scenarios where the two platforms can cooperate to the benefit of both user communities. We believe that this work is a key first step to identifying pieces that could be shared by the two communities, pieces that are complementary, and how Globus and PlanetLab might ultimately evolve together.

We conclude by noting 'experiences' that could potentially be ported from one system to the other:

PlanetLab: Promote interoperability between services. As PlanetLab moves towards a production platform and as the number of services grows, it is likely that both client applications and PlanetLab services will need to interoperate with other PlanetLab services. To facilitate this interoperation, PlanetLab should provide guidelines for service interoperability. We believe that OGSA work on uniform service discovery, representation, invocation, error notification, and the like is a good starting point. Uniform handling of user/service identities and authentication are necessary at a minimum.

PlanetLab: Add support for identity delegation. Currently PlanetLab does not provide identity delegation mechanisms. While these mechanisms can be implemented if needed by higher-level services, there is a risk of ending up with incompatible services. Proxy certificates [57] and GSI offer a possible model.

Globus: Add support for delegating resource usage rights—and address virtualization. The PlanetLab resource usage delegation approach rests on many technical advances that came about after the Globus project started, but that are not yet complete. PlanetLab relies on kernel functions for fine-grained resource control, which are well-developed in the research community (e.g., Scout [52] and resource containers [76]) but shockingly weak in deployed systems. Most current Globus compatible resource schedulers employ identity delegation only. As fine-grained resource control technologies mature and gain deployment, the WS-Agreement protocol can be used as vehicle to experiment with global schedulers based on delegating the right to consume resources, building on PlanetLab experience using SHARP [2] and other global schedulers. WS-Agreement can also provide a basis for negotiating other aspects of resource virtualization, such as installed software and network connectivity.

Globus: Integrating community contributions. PlanetLab's setup as a single VO enables an effective feedback loop to integrate community contributions. User contributions appear first as *service deployments*, which, if proven successful, can be integrated in the testbed infrastructure if they are considered 'public goods', or continue to live as independent services provided an appropriate solution to compensate their providers is found. For grids, user contributions may take a similar path at the VO level. However, this limits the impact user contributions have on the grid community as a whole. Contributions to the entire Globus Toolkit are generally *service implementations* – arguably with a more laborious adoption and deployment process. One possible solution to streamline the adoption process of user contributions for Globus is to enable VOs to outsource non-critical services to a PlanetLab-style backbone.

Finally, we note that both Globus and PlanetLab face significant challenges as they seek to construct open but secure distributed systems in an increasingly hostile Internet. There will surely be advantages to pooling experiences and expertise as the two communities attack critical security and policy issues.

7. Acknowledgements

We are grateful to Robert Adams, Paul Brett, Lenitra Clay, Adriana Iamnitchi, Anne Rogers, and Vijay Tewary for insightful discussions and support. Matei Ripeanu was an intern with Intel Research during the initial stages of this research.

8. References

[1] I. Foster and C. Kesselman, "Globus: A Metacomputing Infrastructure Toolkit," *International Journal of Supercomputer Applications*, vol. 11, pp. 115-128, 1997.

[2] Y. Fu, J. Chase, B. N. Chun, S. Schwab, and A. Vahdat, "SHARP: An Architecture for Sercure Resource Peering," ACM 19th Symposium on Operating Systems Principles, Lake George, NY, 2003.

[3] I. Foster, C. Kesselman, J. Nick, and S. Tuecke, "The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration," Globus Project 2002.

[4] I. Foster, C. Kesselman, G. Tsudik, and S. Tuecke, "A Security Architecture for Computational Grids," *ACM Conference on Computers and Security*, 1998, pp. 83-91.

[5] K. Czajkowski, A. Dan, J. Rofrano, S. Tuecke, and M. Xu, "WS-Agreement: Agreement-based Grid Service Management," *Global Grid Forum*, 2003.

[6] "Global Grid Forum http://www.grid-forum.org," 2002.

[7] I. Foster and C. Kesselman, *The Grid: Blueprint for a New Computing Infrastructure (Second Edition)*, Morgan-Kaufmann, 2004.

[8] C. Catlett and L. Smarr, "Metacomputing," *Communications of the ACM*, vol. 35, pp. 44--52, 1992.

[9] W. E. Johnston, D. Gannon, and B. Nitzberg, "Grids as Production Computing Environments: The Engineering Aspects of NASA's Information Power Grid," 8th IEEE Symposium on High Performance Distributed Computing (HPDC-8), 1999.

[10] G. Allen, T. Dramlitsch, I. Foster, T. Goodale, N. Karonis, M. Ripeanu, E. Seidel, and B. Toonen, "Supporting Efficient Execution in Heterogeneous Distributed Computing Environments with Cactus and Globus," SC'2001, Denver Colorado, 2001.

[11] J. Beiriger, W. Johnson, H. Bivens, S. Humphreys, and R. Rhea, "Constructing the ASCI Grid," 9th IEEE Symposium on High Performance Distributed Computing (HPDC-9), 2000.

[12] I. Foster, E. Alpert, A. Chervenak, B. Drach, C. Kesselman, V. Nefedova, D. Middleton, A. Shoshani, A. Sim, and D. Williams, "The Earth System Grid II: Turning Climate Datasets Into Community Resources," Annual Meeting of the American Meteorological Society, 2002.

[13] C. Catlett, "The TeraGrid: A Primer," www.teragrid.org, 2002.
[14] L. Peterson, T. Anderson, D. Culler, and T. Roscoe, "A Blueprint for Introducing Disruptive Technology into the Internet," ACM HotNets-I Workshop, Princeton, NJ, 2002.
[15] A. Bavier, M. Bowman, B. Chun, D. Culler, S. Karlin, S. Muir, L. Peterson, T. Roscoe, T. Spalink, and M. Wawrzoniak, "Operating System Support for Planetary-Scale Network Services," NSDI'04, San Francisco, CA, 2004.
[16] L. Wang, V. Pai, and L. Peterson, "The Effectiveness of Request Redirection on CDN Robustness," 5th Symposium on Operating Systems Design and Implementation (OSDI'02), Boston, MA, 2002.
[17] N. Feamster, M. Balazinska, G. Harfst, H. Balakrishnan, and D. Karger, "Infranet: Circumventing Censorship and Surveillance," 11th USENIX Security Symposium, San Francisco, CA, 2002.
[18] Y.-H. Chu, S. G. Rao, S. Seshan, and H. Zhang, "A Case for End System Multicast," *IEEE Journal on Selected Areas in Communication (JSAC), Special Issue on Networking Support for Multicast*, vol. 20, 2002.
[19] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris, "Resilient Overlay Networks," 18th ACM Symposium on Operating Systems Principles, Banff, Canada, 2001.
[20] N. Spring, D. Wetherall, and T. Anderson, "Scriptroute: A Public Internet Measurement Facility," USENIX Symposium on Internet Technologies and Systems, 2003.
[21] R. Wolski, "Forecasting Network Performance to Support Dynamic Scheduling Using the Network Weather Service," 6th IEEE Symposium on High Performance Distributed Computing, Portland, OR, 1997.
[22] I. Pratt, D. McAuley, and S. Hand, "PlanetProbe (http://www.cl.cam.ac.uk/Research/SRG/netos/)," 2003.
[23] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications," SIGCOMM 2001, San Diego, USA, 2001.
[24] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A Scalable Content-Addressable Network," SIGCOMM 2001, San Diego USA, 2001.
[25] A. Rowstron and P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems," IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), Heidelberg, Germany, 2001.
[26] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph, "Tapestry: An infrastructure for fault-tolerant wide-area location and routing," UC Berkeley, Technical Report CSD-01-1141, 2001.
[27] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao, "OceanStore: An Architecture for Global-Scale Persistent Storage," 9th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2000), Cambridge, MA, 2000.
[28] M. Castro, P. Druschel, A. M. Kermarrec, and A. Rowstron, "SCRIBE: A large-scale and decentralised application-level multicast infrastructure," *IEEE Journal on Selected Areas in Communications (JSAC) (Special issue on Network Support for Multicast Communications)*, 2002.
[29] H. Yu and A. Vahdat, "Design and Evaluation of a Conit-based Continuous Consistency Model for Replicated Services," *ACM Transactions on Computer Systems (TOCS)*, 2002.
[30] P. Avery and I. Foster, "The GriPhyN Project: Towards Petascale Virtual Data Grids," GriPhyN-2001-15, 2001.
[31] "The DataGrid Architecture," EU DataGrid Project DataGrid-12-D12.4-333671-3-0, 2001.
[32] "Particle Physics Data Grid Project (PPDG), www.ppdg.net."
[33] P. Avery, I. Foster, R. Gardner, H. Newman, and A. Szalay, "An International Virtual-Data Grid Laboratory for Data Intensive Science," Technical Report GriPhyN-2001-2, 2001.
[34] J. Annis, Y. Zhao, J. Vöckler, M. Wilde, S. Kent, and I. Foster, "Applying Chimera virtual data concepts to cluster finding in the Sloan Sky Survey," SC'02, 2002.
[35] L. Childers, T. Disz, R. Olson, M. E. Papka, R. Stevens, and T. Udeshi, "Access Grid: Immersive Group-to-Group Collaborative Visualization," 4th International Immersive Projection Technology Workshop, 2000.
[36] C. Kesselman, T. Prudhomme, and I. Foster, "Distributed Telepresence: The NEESgrid Earthquake Engineering Collaboratory," in *The Grid: Blueprint for a New Computing Infrastructure (2nd Edition)*, I. Foster, Ed.: Morgan Kaufmann, 2004.
[37] T. DeFanti and R. Stevens, "Teleimmersion," in *The Grid: Blueprint for a New Computing Infrastructure*, I. Foster and C. Kesselman, Eds: Morgan Kaufmann, 1999, pp. 131-155.
[38] F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica, "Wide-area cooperative storage with CFS," 18th ACM Symposium on Operating Systems Principles (SOSP '01), Chateau Lake Louise, Banff, Canada, 2001.
[39] A. Muthitacharoen, R. Morris, T. M. Gil, and B. Chen, "Ivy: A Read/Write Peer-to-peer File System," Fifth Symposium on Operating Systems Design and Implementation (OSDI'02), Boston, MA, 2002.
[40] K. Fu, M. F. Kaashoek, and D. Mazières, "Fast and secure distributed read-only file system," *ACM Transactions on Computer Systems*, vol. 20, pp. 1-24, 2002.
[41] R. Huebsch, J. M. Hellerstein, N. Lanham, B. T. Loo, S. Shenker, and I. Stoica, "Querying the Internet with PIER," 30th International Conference on Very Large Data Bases (VLDB'03), 2003.
[42] M. Wawrzoniak, L. Peterson, and T. Roscoe, "Sophia: An Information Plane for Networked Systems," PDN-03-014, June 2003.
[43] P. B. Gibbons, B. Karp, Y. Ke, S. Nath, and S. Seshan, "IrisNet: An Architecture for a World-Wide Sensor Web," *IEEE Pervasive Computing.*, pp. 22-33, 2003.
[44] Ganglia, http://ganglia.sourceforge.net/, 2001.
[45] J. Touch, "Dynamic Internet Overlay Deployment and Management Using the X-Bone," *Computer Networks*, vol. 36, pp. 117-135, 2001.
[46] L. Pearlman, C. Kesselman, S. Gullapalli, B. F. Spencer, J. Futrelle, K. Ricker, I. Foster, P. Hubbard, and C. Severance, "Distributed Hybrid Earthquake Engineering

Experiments: Experiences with a Ground-Shaking Grid Application," 13th IEEE International Symposium on High Performance Distributed Computing, Honolulu (HPDC-13), HA, 2004.
[47] M. Ellisman and S. Peltier, "Medical Data Federation: The Biomedical Informatics Research Network," in *The Grid: Blueprint for a New Computing Infrastructure (2nd Edition)*, I. Foster and C. Kesselman, Eds.: Morgan Kaufmann, 2004.
[48] B. Chun and T. Spalink, "Slice Creation and Management," PDN-03-13, June 2003.
[49] I. Foster, C. Kesselman, and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," *International Journal of High Performance Computing Applications*, vol. 15, pp. 200-222, 2001.
[50] K. Keahey, M. Ripeanu, and K. Doering, "Dynamic Creation and Management of Runtime Environments in the Grid," GGF Workshop on Designing and Building Grid Services, Chicago, IL, October 2003.
[51] M. Gasser and E. McDermott, "An Architecture for Practical Delegation in a Distributed System," IEEE Symposium on Research in Security and Privacy, 1990.
[52] A. Bavier, T. Voigt, M. Wawrzoniak, L. Peterson, and P. Gunningberg, "SILK: Scout Paths in the Linux Kernel," TR 2002-009, Department of Information Technology, Uppsala University, Uppsala, Sweden 2002.
[53] K. Czajkowski, I. Foster, and C. Kesselman, "Resource and Service Management," in *The Grid: Blueprint for a New Computing Infrastructure*, I. Foster and C. Kesselman, Eds.: Morgan Kaufmann, 2004.
[54] L. Pearlman, V. Welch, I. Foster, C. Kesselman, and S. Tuecke, "A Community Authorization Service for Group Collaboration," IEEE 3rd International Workshop on Policies for Distributed Systems and Networks, 2002.
[55] R. Butler, D. Engert, I. Foster, C. Kesselman, S. Tuecke, J. Volmer, and V. Welch, "Design and Deployment of a National-Scale Authentication Infrastructure," *IEEE Computer*, vol. 33, pp. 60-66, 2000.
[56] V. Welch, F. Siebenlist, I. Foster, J. Bresnahan, K. Czajkowski, J. Gawor, C. Kesselman, S. Meder, L. Pearlman, and S. Tuecke, "Security for Grid Services," 12th International Symposium on High Performance Distributed Computing (HPDC-12), 2003.
[57] V. Welch, I. Foster, C. Kesselman, O. Mulmo, L. Pearlman, S. Tuecke, J. Gawor, S. Meder, and F. Siebenlist, "X.509 Proxy Certificates for Dynamic Delegation," presented at 3rd Annual PKI R&D Workshop, Gaithersburg, MD, 2004.
[58] S. Tuecke, D. Engert, I. Foster, M. Thompson, L. Pearlman, and C. Kesselman, "Internet X.509 Public Key Infrastructure Proxy Certificate Profile," IETF draft, 2001.
[59] J. Chase, D. Irwin, L. Grit, J. Moore, and S. Sprenkle, "Dynamic Virtual Clusters in a Grid Site Manager," 12th IEEE International Symposium on High Performance Distributed Computing (HPDC-12), 2003.
[60] D. Abramson, R. Sosic, J. Giddy, and B. Hall, "Nimrod: A Tool for Performing Parameterised Simulations Using Distributed Workstations," 4th IEEE Symposium on High Performance Distributed Computing, 1995.
[61] H. Dail, O. Sievert, F. Berman, H. Casanova, A. YarKhan, S. Vadhiyar, J. Dongarra, C. Liu, L. Yang, D. Angulo, and I. Foster, "Scheduling in the Grid Application Development Software Project," in *Resource Management in the Grid*: Kluwer, 2003.
[62] The Condor Project, http://www.cs.wisc.edu/condor/.
[63] S. Vadhiyar and J. Dongarra, "Metascheduler for the Grid," 11th IEEE International Symposium on High Performance Distributed Computing (HPDC-11), Edinburgh, Scotland, 2002.
[64] K. Czajkowski, I. Foster, and C. Kesselman, "Co-allocation Services for Computational Grids," 8th IEEE Symposium on High Performance Distributed Computing (HPDC-8), 1999.
[65] "Community Scheduling Framework (CSF), http://www.platform.com/," 2003.
[66] 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 Data Sets," *J. Network and Computer Applications*, pp. 187-200, 2001.
[67] "Globus Toolkit - Reliable File Transfer Service" www-unix.globus.org/toolkit/reliable_transfer.html, 2003.
[68] W. Allcock, J. Bester, J. Bresnahan, A. Chervenak, I. Foster, C. Kesselman, S. Meder, V. Nefedova, D. Quesnel, and S. Tuecke, "Data Management and Transfer in High-Performance Computational Grid Environments," *Parallel Computing Journal*, 2001.
[69] M. Zhang, J. Lai, A. Krishnamurthy, L. Peterson, and R. Wang, "Improving Performance and Reliability with Multi-Path TCP," unpublished, Princeton University 2004.
[70] H. Tahilramani Kaur, S. Kalyanaraman, A. Wesss, S. Kanwar, and A. Gandhi, "BANANAS: An Evolutionary Framework for Explicit and Multipath Routing in the Internet," SIGCOMM FDNA Workshop, 2003.
[71] J. Frey, T. Tannenbaum, I. Foster, M. Livny, and S. Tuecke, "Condor-G: A Computation Management Agent for Multi-Institutional Grids," 10th IEEE International Symposium on High Performance Distributed Computing (HPDC-10), San Francisco, California, 2001.
[72] D. Thain, J. Basney, S.-C. Son, and M. Livny, "The Kangaroo Approach to Data Movement on the Grid," 10th IEEE International Symposium on High Performance Distributed Computing (HPDC-10), San Francisco, California, 2001.
[73] K. Ranganathan and I. Foster, "Decoupling Computation and Data Scheduling in Distributed Data-Intensive Applications," 11th IEEE International Symposium on High Performance Distributed Computing (HPDC-11), Edinburgh, Scotland, 2002.
[74] T. Kosar and M. Livny, "Stork: Making Data Placement a First Class Citizen in the Grid," 24th IEEE Int. Conference on Distributed Computing Systems (ICDCS2004), Tokyo, Japan, 2004.
[75] The_Grid2003_Project, "The Grid3 Production Grid: Principles and Practice, Robert Gardner," 13th IEEE International Symposium on High Performance Distributed Computing,, Honolulu, HA, 2004.
[76] G. Banga, P. Druschel, and J. C. Mogul, "Resource containers: A new facility for resource management in server systems," 5th Symposium on Operating Systems Design and Implementation (OSDI-5), 1999.