The Perl Commodity Grid Toolkit

Stephen Mock, Mary Thomas
University of California at San Diego, San Diego Supercomputer Center
{mock, mthomas, mdahan}@sdsc.edu
Gregor von Lazewski
Argonne National Lab
gregor@mcs.anl.gov

1. Introduction

The Computational Grid is the term applied to the infrastructure being constructed to coordinate the use of widely distributed computational resources for high end problem solving. Complex services are deployed to add functionality like authentication, remote access to resources, resource management, and directory services to the Grid. The development of the Grid has been focused on high-end computing, rather than user interface. Using these complex services provides challenges to users and developers of Grid software.

Web portals like the NPACI HotPage [1] simplify the use of Grid services by hiding the complexities of the Grid from portal users while giving them powerful tools accessible through an easy-to-use interface. Without a toolkit that provides interfaces to the Grid services, the developers of each new portal must master the complexities of the Grid in order to incorporate the Grid services into their software. The Commodity Grid project is working to overcome these difficulties by creating what are called Commodity Grid Toolkits (CoG Kits) [2]. "A Commodity Grid Toolkit (CoG Kit) defines and implements a set of general components that map Grid functionality into a commodity environment/framework." [3] The Commodity Grid project is developing CoG Kits in Java, Python, CORBA, and Perl. The Perl CoG Kit, like the others, provides an API to Grid services for developers of higher-level Grid applications.

The Perl CoG Kit is intended to be used by any developer who wants to use the Perl programming language, and has a need to use Grid services in the application that they are developing. In particular, web portals are a good example of applications that use grid services, although there are any number of other example applications which fall into this category.

The NPACI HotPage is a portal that utilizes the Grid to provide users with web access to computational resources at NPACI [4]. The HotPage began as a portal to give users information about resources across the NPACI organization. The need to add interactive features like file manipulation and job submission provided the motivation for creating software to help interface the portal and the Grid. The code that was developed to give HotPage access to Grid services, like job submission, was packaged into a toolkit called the Grid Portal Toolkit, or GridPort [5]. GridPort supports the development of web based Grid portals. It provides access to authentication, job submission and management, and file manipulation via a Perl library interface, as shown in figure 1. A number of portals, including the Telescience Portal [6], LAPK [7], and GAMESS [8] use GridPort. GridPort is a set of Perl libraries, rather than Perl modules, and it does not offer object-oriented interfaces to any of its code. Our experiences in building GridPort were very useful in creating the Perl CoG.

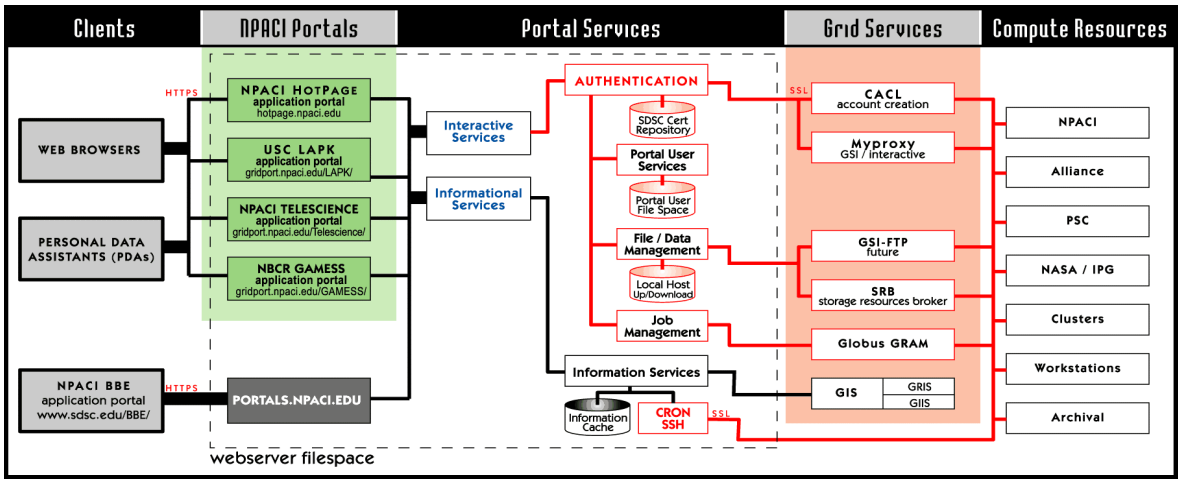

Figure 1. The diagram shows the layered architecture used by the portals based on the GridPort toolkit. This same architecture applies to a portal developed with the Perl CoG Kit.

2. Why Perl

Perl is a useful and versatile programming language. It can be installed on almost any operating system, and most Unix and Unix-like operating systems come with Perl installed as part of the standard installation. Consequently, Perl is a highly portable language used for everything from system utilities to web applications. In particular, the power of Perl for parsing text has made it popular as a web scripting language and has given it widespread use by everyone from scientists to web programmers. The system administrator does not need to make extensions to a webserver to allow it to run Perl/CGI scripts. Perl can be object-oriented if a developer so chooses, depending on the complexity of the application being developed. If the programmer is developing a simple script, the overhead will often outweigh the benefits of object oriented programming. However, if the developer is writing a more complex application, the organizational benefits of object oriented programming often make an object oriented approach attractive. In this case, the developer can use the object-oriented features of Perl to keep the code clear and well organized.

3. Requirements

We developed the Perl CoG Kit with Perl version 5.0 or higher to run on Unix and Unix-like systems. In order to install the Perl CoG, Perl version 5.0 or higher must be installed on the target system. The CoG Kit modules use a few other Perl modules, such as Net::LDAP.pm and Expect.pm to accomplish their tasks. The modules used by the Perl CoG Kit are either included with most Perl installations, or are available for free download from CPAN, the Comprehensive Perl Archive Network[9].

Much of the Grid services functionality of the Perl CoG Kit relies on services provided by Globus. The Perl CoG Kit uses its modules to contact the Globus gatekeeper on the remote machines in order to submit jobs, and it uses the Globus MDS to gather information. The Perl CoG Kit also uses the installed Globus [10] client utilities in some cases to make requests to the server side Globus daemons. Some of the modules are wrappers to the binary executables of the globus installation such as ‘globusrun’ and ‘globus-job-submit’. The CoG Kit also uses the installed GSI authentication utilities, like ‘grid-proxy-init’, that come with the Globus client tools to implement much of the authentication functionality. The installation of the Globus client utilities may present some challenge, although the Globus team is making the installation easier with each update.

The installation scripts for some modules query the installer for information about the system, such as where the local Globus installation is located. The Perl CoG Kit installation follows the CPAN [9] conventions of providing a makefile and a test script for each of the modules. The requirements for the Perl CoG are minimal, and the installation is simple.

4. Perl CoG Architecture

The initial list of modules to be created in the Perl CoG Kit arose from collaboration between the Perl CoG developers and members of the Globus Team. Modules were selected that would support the needs of the developers of the Perl CoG Kit, yet also serve the needs of the general developer community. In addition, modules that would provide a way to make a large impact quickly were developed first, so that the Perl CoG Kit could attract more users and generate feedback quickly. Additionally, we intend that this toolkit will contain input from other developers in the Perl community.

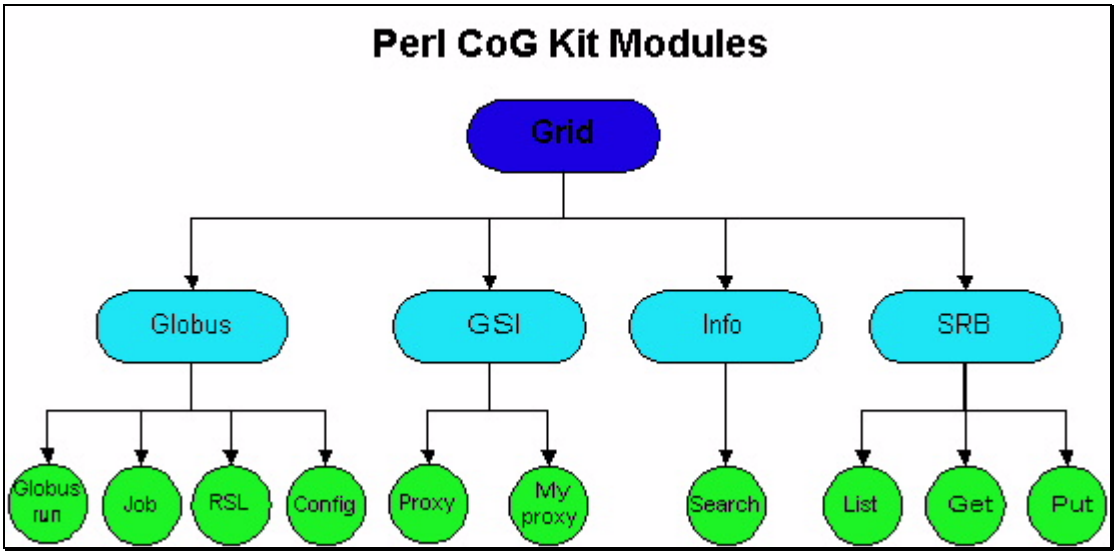

Figure 2. Perl CoG Module Architecture

The architecture of the Perl CoG Kit modules is shown in figure 2. A full description of each module and every subroutine is beyond the scope of this paper. For complete documentation, please refer to the Perl CoG Kit project page [11], or the Perl CoG distribution itself.

The ability for users to submit remote jobs using the Grid is very important. The Perl CoG Kit has modules for dealing with both interactive and batch jobs. The interactive jobs are managed through the Globus::Globusrun module. It allows for the assembly and execution of interactive jobs via Globus GRAM [12], and returns the output of the interactive job. The Globusrun module is designed as a utility module, with no object-oriented features. Handling batch jobs is done in the Perl CoG Kit with the Globus::GlobusJob module. The Globus::Job module is an object-oriented module, in which developers create job objects, and then can set or get the parameters of the job object. When the job has been assembled, it can be submitted. The module provides mechanisms for checking the status of the job, canceling the job, and getting the output of the job. Both the Globusrun and Job modules depend on installed Globus client utilities to implement their functionality. The Perl CoG implements a clean simple interface to job submission through Globus.

The Resource Specification Language (RSL) is the language the Globus software uses as a common language to specify job requests [13]. The Perl CoG contains an RSL module for that provides an interface to create, manipulate, and verify RSL strings. This module provides an object-oriented interface to the creation of RSL strings. It contains subroutines to create and delete RSL objects, add and delete attributes from the RSL objects, and to output RSL strings from the RSL objects. The RSL module is pure Perl, meaning that it does not depend on any outside, non-Perl utilities or libraries to install and operate correctly.

The Config module found in the Globus branch is a module for configuring the CoG interface to the local Globus installation on the system. The installation of the Config module attempts to locate the Globus installation, or prompts the installer for help. This module provides the other modules that use Globus system tools with a simple and uniform access to those installed Globus utilities. It also provides an easy mechanism for the CoG maintainer to switch the installation of Globus that the CoG uses. The Config module is primarily a convenience mechanism for the developers and maintainers of the Perl CoG.

Grid Security Infrastructure (GSI) is emerging as a dominant authentication and security mechanism [14]. The Perl CoG supplies modules for managing GSI credentials through the GSI::Proxy and GSI::Myproxy modules. The Proxy module depends on some of the installed GSI system utilities that are included in the Globus client software installation; thus it is not pure Perl. The Proxy module facilitates the generation of proxy certificates from an X.509 certificate and key pair. The creation of this proxy allows delegation of authority to access remote Grid services. The Proxy module also provides the ability to destroy a proxy, and to get information about a previously existing proxy file such as the subject, strength, type, issuer, and time to live of the delegated credential. In addition to the Proxy module, the Perl CoG Kit supplies the GSI::Myproxy module to support the use of Myproxy[15] for managing GSI credentials. The Myproxy module has subroutines to insert a credential into a Myproxy server, to get a delegated credential from a Myproxy server, and to delete a credential from a Myproxy server. The Myproxy module depends on the installed Myproxy system utilities as well as the Expect Perl module.

Information discovery is handled in the Perl CoG through the Info::Search Perl module. The Info::Search module is an object-oriented module which facilitates the assembly and execution of queries against the Globus MDS [16]. The developer creates a search object by specifying the MDS host and port and setting its parameters, and then executing a search. The search object contains the entries returned, which can be parsed and used to make decisions about further Grid activity. For example, a search could be performed to find a computer with the needed available compute resources for a particular job. The Search module depends on the two Perl modules, the Net::LDAP module, which it uses to make the LDAP connection to the MDS for the queries, and the LDAP::Entry module, which describes the data structure that the results from a search are returned in. The Search module is a pure Perl implementation; the only two requirements for it are Perl modules. The Perl modules it requires are freely available from the CPAN website. [9]

The SRB [17] module is in development at this time, and will be available on the Perl CoG project page when complete [11].

5. Module Creation

The skeleton of each of the modules was created using a utility from the standard Perl installation called h2xs. The h2xs utility builds a Perl extension from any C header file, but also has the ability to simply put together the framework for building a new module. This includes providing a stub module file and testscript, creating a makefile generator, and installing all the necessary files and directories needed to create a module complete with manpages and documentation. The testscript has dual functionality: it supplies the installer with a mechanism for checking that the installation worked correctly, and it gives the developers a working example of how to use the module. So in addition to the Perldoc documentation that comes with each module, the developers can refer to the testscripts for additional examples.

6. Project Status and Future Plans

The Perl CoG Kit is still in the early stages of development. A small subset of the target functionality has been implemented already, though some of the modules already completed are being redesigned. The core functionality of the CoG Kit is in place. Modules for job assembly, batch job submission, interactive job submission, authentication, and information discovery have been completed and tested. The Perl CoG source code is available from the Perl CoG project web page found at http://gridport.npaci.edu/cog.

Since the Perl CoG is a new addition to the CoG Project it has many more modules to be developed. Modules for other Grid services are either being developed now, or will be developed in the

future. We are working on modules to interface with the Storage Resource Broker (SRB) [17] for distributed file collection access, as well as modules for transferring files between remote systems using gsiftp. Some of the modules that require system utilities like the Globus utilities are under examination to determine if they can be implemented as pure Perl, the result of which could be a pure Perl implementation of GRAM. This benefit of taking this approach is that the CoG could be a replacement for the utilities that it provides wrappers for. Since Perl is highly portable, a pure Perl implementation could lead to Grid access from new platforms.

The GridPort toolkit is in the process of being rewritten using the Perl CoG, making GridPort a cleaner and more portable package. The GridPort toolkit itself will include modules for doing authentication and running jobs that are particularly specialized for use with web portals based on the modules of the Perl CoG.

The Perl CoG team is currently evaluating the use of SWIG [18], the Simple Wrapper and Interface Generator, to aid in the development of core Globus functionality like GRAM. The Python CoG Kit has demonstrated successful use of SWIG to interface directly with the Globus C code. This approach allows reuse of the Globus Team's code, while providing the Python language with access to the functionality contained in the Globus code. This provides a good starting point to enable another language, like Python or Perl, access to the software that Globus has already developed. SWIG has the ability to provide this sort of interface in several languages, including Python and Perl.

7. Conclusion

The Perl CoG brings the Grid to the Perl community, enabling more people and applications to become aware of and use Grid services. The potential usefulness of the Perl CoG extends beyond the web portal community: it enables the integration of Grid services into command line and system utilities that are written in Perl. The number of web-based portals under development is enormous, and many of them have a need for Grid services. Perl's easy integration into web-based applications makes it a great candidate for use in making these new portals Grid-aware, and the Perl CoG can make the process significantly easier.

8. References

[1] M. Thomas, S. Mock. The NPACI HotPage User Portal. https://hotpage.npaci.edu.

[2] Commodity Grid Kits Project Web Page. Available at http://www.globus.org/cog.

[3] Gregor von Laszewski, Ian Foster, Jarek Gawor. CoG Kits: A Bridge between Commodity Distributed Computing and High-Performance Grids. Available at: http://www-unix.mcs.anl.gov/~laszewsk/papers/cog-final.pdf

[4] National Partnership for Advanced Computational Infrastructure (NPACI). Project Web Page. http://www.npaci.edu.

[5] The GridPort Toolkit: a System for Building Grid Portals. M. Thomas, S. Mock, M. Dahan, K. Mueller, D. Sutton, J. Boisseau. *10th IEEE Symp. on High Performance Distributed Computing.* Aug. 2001.

[6] The Telescience Web Portal. https://gridport.npaci.edu/Telescience/.

[7] The Laboratory of Applied Pharmacokinetics Web Portal. https://gridport.npaci.edu/LAPK.

[8] The General Atomic Molecular Electronic Structure Systems (GAMESS) Web Portal.

https://gridport.npaci.edu/GAMESS.

[9] The Comprehensive Perl Archive Network.
http://www.cpan.org.

[10] The Globus Project Web Page.
http://www.globus.org.

[11] The Perl CoG Kit Project Web Page.
https://gridport.npaci.edu/cog.

[12] The Grid Resource Allocation Manager.
http://www.globus.org/gram/.

[13] The Globus Resource Specification Language.
Available at: http://www.globus.org/gram/rsl.html.

[14] I. Foster, C. Kesselman, G. Tsudik, S. Tuecke. A Security Architecture for Computational Grids. *Proc. 5th ACM Conference on Computer and Communications Security Conference*, pg. 83-92, 1998.

[15] Myproxy (v 1.0).
Available at: http://dast.nlanr.net/Projects/MyProxy.

[16] The Metacomputing Directory Service.
http://www.globus.org/mds/.

[17] The Storage Resource Broker.
http://www.npaci.edu/Thrusts/DI/SRB/.

[18] SWIG, Simplified Wrapper and Interface Generator.
http://www.swig.org.