GROWL Scripts: Lightweight Access to Grid Resources

John Kewley, Adam Braimah and Rob Allan
Science & Technology Facilities Council
Daresbury Laboratory e-Science Centre
r.j.allan@dl.ac.uk and j.kewley@dl.ac.uk

Mark Hayes and Peter Brorsson
Cambridge eScience Centre, University of Cambridge
mah1002@cam.ac.uk and pb396@hermes.cam.ac.uk

Rob Crouchley, Daniel Grose and Ties van Ark
Centre for e-Science, University of Lancaster
r.crouchley@lancaster.ac.uk, d.grose@lancaster.ac.uk and t.vanark@lancaster.ac.uk

Abstract

There are currently many people moving from running jobs on clusters to running jobs on the Grid using a Globus command line interface. They typically don't find this transition as simple as they should: they have problems building the software, have to learn about X509 certificate manipulation, and run into problems with firewalls. As well as providing a Web Service interface, GROWL addresses these problems by providing wrapper scripts for Globus and associated tools. While not being as generally applicable as the GROWL Web Services due to some minor firewall restrictions, GROWL Scripts are still being used in a variety of projects to ease the transition from cluster to Grid.

1 Background

There are many scientists using clusters as a vital tool to perform their scientific work. The promise of true Grid computing should be highly attractive to such scientists. There are however many who would shun the current plethora of customised Grid portals in favour of an effective command line interface. Indeed this is what was first provided by the Globus team and is in widespread use. There are however several barriers which are frequently encountered in its use by scientists who are accustomed to the interfaces provided by cluster computing.

Firstly, there is the problem of obtaining the correct client middleware. To be able to run Grid jobs and copy files to and from Grid resources, software from several sources is needed (for example Globus, gsi-enabled OpenSSH, myproxy and various patches that must be applied to these). This was considerably simplified with the approach taken by the Virtual Data Toolkit (VDT), so that the middleware is obtained from a single location and you only need to download the components you require. Although significantly reducing the effort of installing Grid middleware, the commands to set-up the package manager used by VDT (`pacman`) are still non-trivial and unless you know exactly which packages to request, you can end up with more than you need. Another problem here is that VDT isn't available on as large a range of platforms as Globus is. This is because VDT contains a wider range of middleware and therefore supports only those that are common.

Secondly, these computational scientists are typically unfamiliar with X509 Grid certificates and the commands that are used to manipulate them. Instead of logging in to a remote machine using `ssh` before submitting their batch jobs, they typically have to use `gsissh`. This entails first obtaining a Grid certificate by application from the appropriate authority (for UK e-Science this is the UK e-Science Certification Authority [1]. Once this is received, they then must ensure it is on the correct machine(s) and in the right format (usually `.pem`, although some environments permit leaving it in `.p12` format). This has to be repeated each year when they renew their certificate. Now, each time before logging in to the remote machine they must ensure they have a valid proxy certificate for their session; this is achieved through the use of either `grid-proxy-init` or `myproxy-init`.

Thirdly, there is the issue of firewalls. If the users try to use the Globus middleware to submit batch jobs from their host machine (rather than logging on to the Grid resource), they will find that although their submission is successful, retrieving the results will be problematic since their site or machine firewall may well block the incoming connections coming back from the Grid resource. The same will be true for interactive jobs. This encourages them to logon directly to the remote machine and call the underlying resource managers directly, rather than use the higher level interface provided by Globus.

Some of these issues have previously been raised by Chin and Coveney [2], and Gabriel [3] and a number of projects are now beginning to address these issues; in the UK they include AHE [4] and GROWL [5, 6], the latter was funded by the JISC under its VRE 1 Programme [7] from 2005–07.

1.1 The GROWL Project

The GROWL Project [8] set out to try and alleviate these barriers to Grid adoption by the use of Web Services and a three-tiered architecture as described below. This enabled users to run Grid jobs from within C, C++, Fortran and R programs.

Since there was not initially a method for transferring large files as a Web Service (this functionality is currently being added), shell scripts were written that wrapped `gsiscp` and could be used to transfer such files, avoiding the transfer of non-ASCII data in SOAP. This also avoided the copying of files from client to GROWL Server from where they needed to be further copied again to the Grid resource. To these were added routines to list and remove files and directories on the Grid Resource. C wrappers were provided so they could be integrated with the Web Service submission mechanism and a `growl-login` script written which encapsulated both `grid-proxy-init` and `myproxy-init`; this was because to use both Web Services and `gsiscp`, proxy certificates must be on both a MyProxy [9] Server (so that it can be requested from the central GROWL Server) and on the local machine.

Once this range of scripts were produced, it soon became obvious that the only item missing from the scripts to be able to provide a fully working environment was a submission mechanism. This addition led to the formation of the GROWL Scripts as a distinct entity to the Web Services part of GROWL (whose architecture is described in subsection 2.1 below).

1.2 Web Services

A Web Service is an application accessible using standard Internet protocols. Web Services represent black-box functionality that can be reused without concern for how that service is implemented, or what language it is written in and are accessed via ubiquitous Web protocols (e.g. HTTP) and data formats such as SOAP, WSDL and XML.

GROWL uses gSOAP [10] to generate the client side of the Web Services which are then wrapped by the GROWL API. The gSOAP compiler tools simplify the development of Web services by their provision of C and C++ language bindings for stub and skeleton code generation. This results in a flexible framework that can be also be utilised from Fortran. This would allow a subsequent version of GROWL to provide a Fortran library rather than providing Fortran wrappers to the GROWL C library. The gSOAP stub compiler automatically does all the data conversion from user-defined C and C++ data types to equivalent XML data types and vice-versa.

The client Web Service calls are depicted in the top left of Figure 1 as the calls from **WS I/F** to the GROWL Server.

2 The GROWL Scripts

2.1 Architecture of GROWL

GROWL was initially designed to target Grid application developers and tool developers and is envisaged as a toolkit to write Grid-enabled applications. It uses standard middleware from Grid developers (such as Globus) installed on a server, so addresses issues of security, Grid connectivity and protocols and it enables development of rich Grid-enabled applications to satisfy the require-

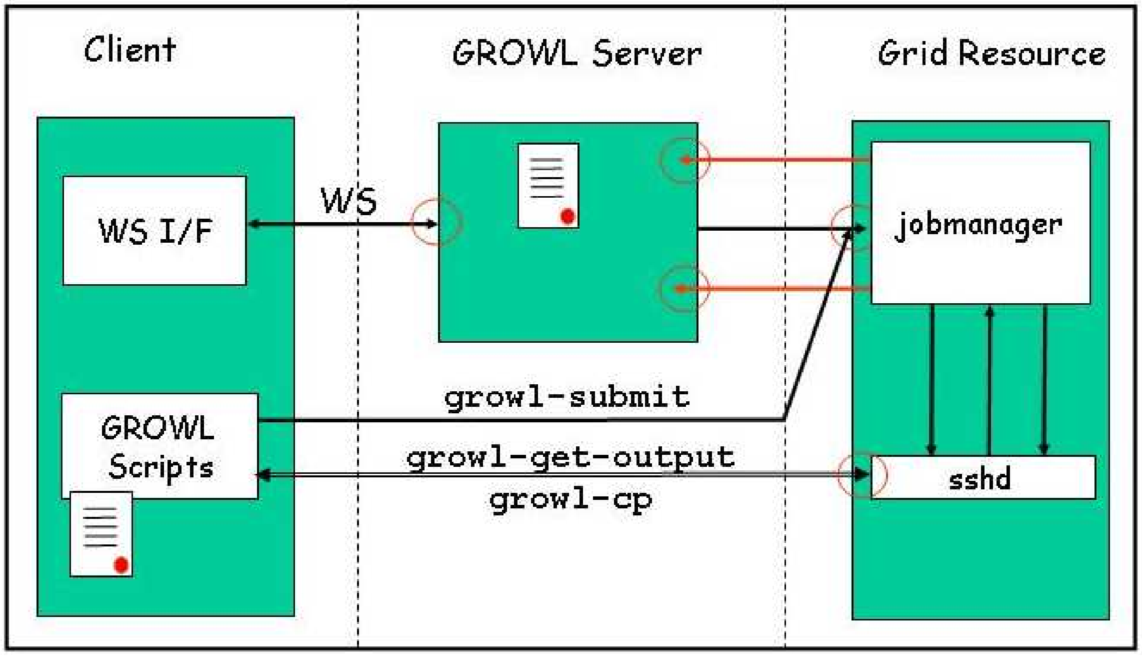

Figure 1: GROWL Web Services and Scripts

ments of end users in carrying out their research.

GROWL has one or more servers which are effectively "part of the Grid". They are secure systems with latest software patches and maintained by system administrators. The functionality of the server is to be addressed by developers in the GROWL project. Application developers access the server or connect directly to remote Grid resources through the GROWL client toolkit which is discussed in this paper. HPC applications and services are installed on Grid resources and maintained by system administrators responsible for those resources.

It is well known that such as three-tier architecture is an appropriate solution for isolating implementation from client and service provider. In addition, it serves to minimize the amount of information each resource requires about other resources and reduces the number of communication pathways. This prevents the growth of an unmanageable "stove pipe" solution that is non scalable. However, given the large number of Grid based applications and the extent of the development community required to develop them, such an architecture needs to be provided *a priori* to the application development itself. Thus, in the first instance, it is necessary to develop three tier frameworks with provide easy to use, open APIs for the developer community. Such technologies already exist, such as CORBA, Web Services and many others. However, none of these, on an individual basis, fully take into account the following requirements which seem basic to a three tier architecture for the Grid:

- Multiple service implementations provided through a common interface where the interface reflects the context of the problem domain, not the specifics of the Grid middleware;
- Secure communication;
- Client identification;
- Client lifetime is different form service lifetime, thus services have to be stateful to be able to efficiently and effectively manage job creation, submission, retrieval and lifetime;
- Clients can only communicate through a limited number of ports;
- Common format for communication with mappings to a number of different languages.

For GROWL we have chosen to use Web Services for communications between the client and server, and Grid protocols (such as Globus) for communications between server and remote resource. Firewalls permitting, direct communication between client and remote resource is also possible using protocols such as GSI-SSH which are relatively straightforward to install and fit with GROWL's "lightweight middleware" philosophy. The Web Service calls to the GROWL Server and their ensuing calls to the Grid protocols on the Grid Resource are shown in the upper part of Figure 1 while the direct calls from the Client to the Grid Resource using GROWL Scripts are shown in the lower half.

2.2 Download and Build

Once GROWL Scripts have been downloaded from the GROWL website [8], they must be first unpacked using `tar -zxvf`. This initial download comprises the build mechanisms for GROWL Scripts (and for GROWL Web Services) and the scripts themselves, the Grid middleware still has to be downloaded and then built and installed. Two alternative methods for this are provided.

The first uses VDT and involves downloading and installing the package manager `pacman` and the VDT packages containing the Globus Client and other software used by the GROWL Scripts. In this case ready-built software is installed using the following command:

```
$ make VDT
$
```

The alternative approach is appropriate for platforms for which VDT is not supported. Note one key difference here is that the downloaded software must be obtained from several different locations on the Grid in source form and then built on the client machine. This method therefore takes considerably longer and uses an alternative `make` target:

```
$ make noVDT
$
```

Note that before using any of the middleware utilities installed above or any of the GROWL Scripts mentioned below, `~/Growl/setup.sh` must be "sourced" to setup the user's environment. There is an analogous script `~/Growl/setup.csh` for users of `csh` and `tcsh`.

2.3 Certificate Handling

In order to use a Grid X509 certificate in many environments it is necessary to convert it from the `.p12` (or `.pfx` for Windows) format to `.pem` using the cryptographic toolkit OpenSSL. This can be used to split a single `.p12` file into the two files `usercert.pem` (the user's public certificate) and `userkey.pem` (the user's private key, pass-phrase protected). These files need to be placed in an appropriate place (typically `.globus` in the user's home directory) and have some restrictive file permissions placed on them. This needs to be done once per year.

The `openssl` and `chmod` commands to do this are as follows (note that `<Pass1>` is the pass-phrase used to encrypt `usercred.p12` and `<Pass2>` is the new pass-phrase to encrypt `userkey.pem`.):

```
$ openssl pkcs12 -in usercred.p12 \
    -nocerts -out userkey.pem
Import Password: <Pass1>
PEM passphrase: <Pass2>
Verifying: PEM passphrase: <Pass2>

$ openssl pkcs12 -in usercred.p12 \
    -clcerts -nokeys -out usercert.pem
Import Password: <Pass1>

$ chmod 400 userkey.pem
$ chmod 444 usercert.pem
$
```

Note how the pass-phrase `<Pass1>` needs to be provided to firstly extract the private key and then again for producing the public certificate. Note also how the `openssl` commands are non-trivial.

GROWL Scripts provide a utility to wrap all of this into one operation.

```
$ mk-cert usercred.p12
Passphrase: usercred.p12: <Pass1>
Passphrase: userkey.pem [same]: <Pass2>
$
```

`mk-cert` permits the same password as was used to encrypt the `.p12` file to be used to reencrypt `userkey.pem`, or alternatively a different one can be provided.

The utility `growl-info` can be used to confirm that your certificate is still valid. It will also check for valid local proxy certificates and proxies uploaded to a default MyProxy server.

```
$ growl-info
Certificate Information (including validity)
--------------------------------------------
subject=/C=UK/O=eScience/.../CN=john kewley
notBefore=Jun 15 16:10:35 2006 GMT
notAfter=Jun 15 16:10:35 2007 GMT

Local proxy certificate(s)
--------------------------
subject : /C=UK/.../CN=john kewley/...
issuer  : /C=UK/.../CN=john kewley
identity: /C=UK/.../CN=john kewley
type    : Proxy draft (pre-RFC) ...
strength: 512 bits
path    : /tmp/x509up_u13445
timeleft: 10:52:01

MyProxy proxy certificate(s)
----------------------------
username: jmk27
owner: /C=UK/O=eScience/.../CN=john kewley
timeleft: 41:15:13  (1.7 days)
$
```

2.4 Filestore manipulation

To provide all the functionality needed for basic work-flows, GROWL Scripts provides a variety of scripts which can be used to manipulate the files on the Grid resource without having to logon to the machine. File and directory removal, copying, renaming and listing are provided along with some other utilities. These work in a similar way to their Unix/Linux counterparts but take a compulsory initial parameter which is the hostname of the Grid resource (a future version of these scripts will also support caching this hostname for simplicity).

Provided scripts are `growl-ls`, `growl-rm`, `growl-mkdir`, `growl-mv` and `growl-cp`. The copying utlity `growl-cp` has some limited support for managing third party copying (i.e., copying to/from a Grid resource from/to another machine which is not the client). When asked to transfer from one Grid resource to another, it tries the following in order, proceeding to the following option if that one fails (note that due to the built-in timeouts in `gsissh` this can take some time if there are firewalls between the two machines):

1. logs on to the first machine and tries to copy from there to the second
2. logs on to the second machine and tries to copy to it from the first machine
3. copies the file to the client from the first machine and then copies it from there to the second machine.

One simple script that has been found useful is `growl-pwd` which simply states the home directory of the user on that machine. This enables you to build up full directory path names of files that you have installed on the Grid resource.

2.5 Job Submission

For instance to submit a job to run hostname to the batch system on `scarf.rl.ac.uk`, the following should be run.

```
$ growl-submit scarf.rl.ac.uk hostname
https://scarf.rl.ac.uk:64001/...
$
```

Note the following:

- The user must have authorisation to run Grid jobs on the machine given as the first of the positional parameters, this requires the user's Distinguished Name to be present in that machine's "gridmap" file (usually `/etc/grid-security/grid.mapfile`) to link it to the account to use for the job;

- The executable as given must exist on the user's path on the remote machine. `growl-submit` will fail if the executable is not in your path on that machine when you submit the job. This can be tested in advance using `growl-which` (as described briefly in paragraph 2.6 below);

- `growl-submit` tries to extract the jobmanager name for the given Grid resource. It will use the first one it finds. If this is not appropriate, you will have to specify the jobmanager when submitting a batch job (see also `growl-get-jobmanager` in paragraph 2.6 below);

- To submit a job to run on the Grid resource head-node, you must append `jobmanager-fork` to the resource name.

2.5.1 Checking the status of a Grid job

`growl-status` is used to check whether a job has completed or not

```
$ growl-status https://scarf.rl.ac.uk...
PENDING

$ growl-status https://scarf.rl.ac.uk...
PENDING

$ growl-status https://scarf.rl.ac.uk...
ACTIVE

$ growl-status https://scarf.rl.ac.uk...
ACTIVE

$ growl-status https://scarf.rl.ac.uk...
DONE

$
```

2.5.2 Retrieving standard output and standard error

Once the job has completed (`DONE`), `growl-get-output` can be used to retrieve either the standard output or standard error of the job as follows:

stdout
```
$ # Get standard output
$ growl-get-output https://scarf.rl...
scarf.rl.ac.uk
$
```

stderr
```
$ # standard error (in this case, empty)
$ growl-get-output -e https://scarf...
$
```

2.6 Miscellaneous

There is an option (`-c`) for `growl-submit` which affects the calls to the other scripts in the preceding subsection. This caches the return string from `growl-submit` in `/tmp`; any subsequent calls to `growl-status` or `growl-get-output` will take this cached value as a default. This is extremely useful for testing purposes when the lines are being typed in by hand and when only one job is current per shell. Due to the way the file is named, it is not robust enough for more complicated use.

There are also other wrapper scripts including the following:

`growl-cancel` and `growl-clean`: used for cancelling and cleaning up after a job.

`growl-get-jobmanager`: queries the Grid resource to try and ascertain which job manager is running so the user need not specify it.

`growl-which`: tests whether a given executable is in your path on the destination machine and, if so, returns where it is located.

`growl-login`: this will ensure that you have both a valid local proxy certificate and one lodged with the default MyProxy server. If both are valid, it does nothing; If the MyProxy certificate is valid, it generates a local proxy certificate from it; otherwise it generates both.

3 In Conclusion

3.1 Benefits

There are several benefits of using GROWL Scripts over the Globus provided scripts directly:

- the facility for `growl-get-output` to retrieve standard output from a Grid resource even if a site or machine firewall prevents incoming connections. This is not possible with `globus-job-get-output`;
- conversion of certificates using `mk-cert`;
- the ability for `growl-submit` to work out the appropriate jobmanager whereas `globus-job-submit` and `globus-submit` need that provided. In fact `growl-submit` defaults to the jobmanager on the Grid resource rather than `jobmanager-fork` as with the Globus commands;
- the transparency offered by `growl-submit` by prefixing the full path to the executable and checking its existence;
- for those who need local proxies and MyProxy proxies, `growl-login` will provide both requesting a minimum of passwords. `growl-logout` and `growl-info` will both operate on both types of proxy;
- the possibility of performing third party file transfers, although note the disclaimers above regarding `ssh` timeouts if there are firewalls between the two Grid resources;
- the caching of the job contact string by `growl-submit` when testing.

3.2 Current Usage

There are currently four usage levels of GROWL Scripts:

1. use of GROWL Scripts to provide an easy to install way of getting the minimum tools to access Grid resources: `grid-proxy-init` and `gsi-ssh`;
2. as above, but also use of `mk-cert` annually to convert certificates;
3. as above, but using a wide set of the Globus middleware directly. This will typically only work if some firewall rules are relaxed. This is the Globus client we install for the back-end of the GROWL Server which is used to handle GROWL Web Services requests;
4. using the file manipulation and job submission wrappers provided by GROWL Scripts from the client machine rather than logging onto the target Grid resource.

Most of our current users are in the first and second categories. At our recent NW-GRID Training event [11, 12], users were shown this method of installing and running Globus software successfully. It has also been used by the CCP1GUI [13] project.

3.3 Restrictions

Note that there are two main caveats when considering whether to use GROWL Scripts or not. Firstly, although the problem of firewalls blocking incoming ports to the Client is effectively removed, there are still the following firewall requirements:

- outgoing ports must be open on the client machine (this is the usual behaviour);
- incoming ports on the Grid resource must be open for `gsissh` and `globus-job-submit` for the client. The latter of these isn't always guaranteed, and indeed some Grids are now restricting access to the former as well.

3.4 Future Work

If this client toolkit is to be successful, it really needs to be available on the Windows platforms. This possibility is currently being evaluated and it is hoped to build upon the work of the Java Commodity Grid (CoG) kit [14] and the GSI-SSHTerm [15] projects so a `gsissh` for Windows can be produced. Note that while GSI-SSHTerm can currently give a terminal window on the remote resource, it does not yet provide a command-line program (`gsissh` drop-in replacement) which can be used to send commands to the remote machine.

References

[1] UK e-Science Certificate Authority (CA)
`ca.grid-support.ac.uk`

[2] Jonathan Chin and Peter Coveney, *Towards tractable toolkits for the Grid: a plea for lightweight, usable middleware*, June 2004,
`www.realitygrid.org/lgpaper21.pdf`

[3] R.P. Gabriel, *The Rise of "Worse is Better"*
`www.jwz.org/doc/worse-is-better.html`

[4] P.V. Coveney, R.S. Saksena, S.J. Zasada, M. McKeown and S. Pickles *The Application Hosting Environment: Lightweight Middleware for Grid-Based Computational Science*, Computational Physics Communications, Volume 176, Issue 6, March 2007, pp 406–18; doi:10.1016/j.cpc.2006.11.011

[5] Mark Hayes, Lorna Morris, Rob Crouchley, Daniel Grose, Ties van Ark, Rob Allan and John Kewley. *GROWL: A Lightweight Grid Services Toolkit and Applications*. In Simon Cox and David W. Walker, editors, Proceedings of the UK e-Science All Hands Meeting 2005. EPSRC, September 2005.
`epubs.cclrc.ac.uk/bitstream/920/460.pdf`

[6] Rob Crouchley, Ties van Ark, John Pritchard, John Kewley, Rob Allan, Mark Hayes, and Lorna Morris. *Putting Social Science Applications on the Grid*. In Proceedings of the First International Conference on e-Social Science. National Centre for e-Social Science, June 2005.
`epubs.cclrc.ac.uk/bitstream/1453/NCeSS.pdf`

[7] JISC VRE Programmes
`www.jisc.ac.uk/index.cfm?name=programme_vre`

[8] The GROWL Project,
`www.growl.org.uk/`

[9] Jim Basney, Marty Humphrey, Von Welch *The MyProxy online credential repository*, Software: Practice and Experience, Volume 35, Issue 9, 2005, pages 801-816, John Wiley & Sons

[10] Robert A. van Engelen and Kyle A. Gallivan *The gSOAP Toolkit for Web Services and Peer-To-Peer Computing Networks* www.cs.fsu.edu/~engelen/soappaper.html

[11] The North West Grid www.nw-grid.ac.uk/

[12] Induction to Grid Computing and the North West Grid www.nw-grid.ac.uk/?q=seminar/dares-250107

[13] J. Thomas, J Kewley, RJ Allan, JM Rintelman, P Sherwood, CL Bailey, S Mukhopadhyay, A Wander, BG Searle, NM Harrison, A Trewinry, GR Darling, AI Cooper *Experiences with different middleware solutions on the NW-GRID*, to appear.

[14] Java Cog Kit wiki.cogkit.org

[15] GSI-SSHTerm www.grid-support.ac.uk/content/view/81/62