glideinWMS experience with glexec

2012 J. Phys.: Conf. Ser. 396 032101

(http://iopscience.iop.org/1742-6596/396/3/032101)

Journal of Physics: Conference Series **396** (2012) 032101 doi:10.1088/1742-6596/396/3/032101

glideinWMS experience with glexec

I Sfiligoi[1], D C Bradley[2], Z Miller[2], B Holzman[3], F Würthwein[1], J M Dost[1], K Bloom[4], C Grandi[5]

[1]University of California San Diego, La Jolla, CA 92093, USA
[2]University of Wisconsin – Madison, Madison, WI 53706, USA
[3]Fermilab, Batavia, IL 60510, USA
[4]University of Nebraska – Lincoln, Lincoln, NE 68588, USA
[5]Istituto Nazionale di Fisica Nucleare - Sezione di Bologna, I-40127 Bologna, Italy

E-mail: isfiligoi@ucsd.edu

Abstract. Multi-user pilot infrastructures provide significant advantages for the communities using them, but also create new security challenges. With Grid authorization and mapping happening with the pilot credential only, final user identity is not properly addressed in the classic Grid paradigm. In order to solve this problem, OSG and EGI have deployed glexec, a privileged executable on the worker nodes that allows for final user authorization and mapping from inside the pilot itself. The glideinWMS instances deployed on OSG have been now using glexec on OSG sites for several years, and have started using it on EGI resources in the past year. The user experience of using glexec has been mostly positive, although there are still some edge cases where things could be improved. This paper provides both the usage statistics as well as a description of the still remaining problems and the expected solutions.

1. Introduction

Many scientific communities, or Virtual Organizations (VOs), have adopted Grid computing as their base computing model to simplify the deployment and management of the tens of thousand of CPUs needed to accomplish their mission. While the Grid computing paradigm has been shown to be a boon for resource providers, allowing them to keep their administrative autonomy over the resources they manage, direct use of these resources has been shown to be difficult for standard users.

As a result, the VOs have adopted the **pilot-based Workload Management System** (WMS) paradigm, also known as “overlay infrastructure”. In this paradigm, resources across multiple administrative domains are aggregated into VO specific overlay pools, or “virtual clusters” (VC), by means of pilot jobs. Each VO has full control over its own VC, and can thus easily implement priorities between the final users. Moreover, resource provisioning is clearly separated from resource usage, with the former managed by dedicated IT personnel. Standard users are thus never exposed to the complexities of Grid infrastructure and perceive the overlay pool as just any other compute cluster.

A major drawback of pilot-based WMS is the **changed security model**. In order to achieve an illusion of a private cluster, the credential used to provision resources at various Grid sites are not

specific to any particular user. A pilot process that joins the overlay pool will accept jobs from any standard user of the pool, and may even run jobs from several users if there is enough time available; this is usually referred to as **multi-user pilot jobs**. This has major security implications for both the trust relationship between the WMS and Grid sites, and the security of the VC itself. In order to address this issue, OSG[1] and EGI[2] have deployed glexec[3], a privileged executable on the worker nodes that allows for final user authorization and mapping from inside the pilot itself.

A broadly adopted multi-user pilot WMS implementation is **glideinWMS**[4], which heavily relies on **Condor**[5] to implement the pilot paradigm. Several OSG VOs, spanning biology, chemistry, climate science, computer science, economics, engineering, mathematics, medicine, and physics, have been using it for several years now. Condor, and by extension glideinWMS, had support for glexec essentially since the day the tool was released, and a few OSG VOs have been using glexec for this whole time, with several others joining later on. In this time span, we have discovered several problems in the integration of Condor with glexec, most of which have been addressed shortly after being discovered.

This paper presents a description of the security challenges of multi-user pilot jobs, the specifics of the integration of Condor with glexec, both as it stands at the time of writing and its historical evolution, as well as a snapshot of how it is currently used by the OSG VOs.

2. The security challenge

The Grid security paradigm is centered around the concept of a one-to-one mapping between physical users and electronic credentials, and all the software involved has been optimized for this use case. Multi-user pilot jobs break this assumption, by using a single credential to execute jobs from many physical users, as shown in Fig. 1.

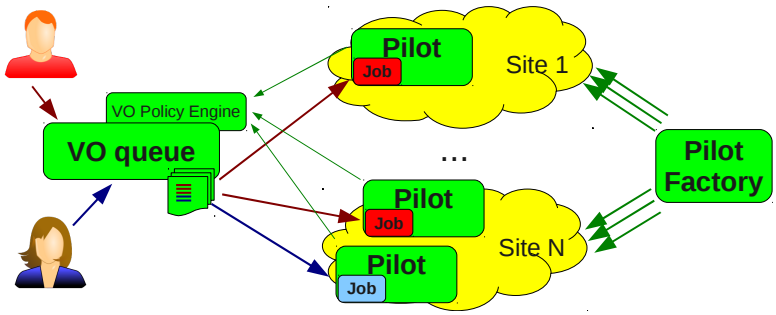

Figure 1. A schematic view of a multi-user pilot WMS

The two main issues stemming from this change involve trust between the pilot WMS and the Grid sites, and the security of the created virtual cluster. This section first describes these two issues, and then explains how glexec addresses them.

2.1. The Grid site trust issue

The Grid is composed of hundreds of independent resource providing sites, each operating in its own administrative domain. A consequence of this premise is that every site retains the right to control who uses their resources.

By submitting multi-user pilot jobs, a multi-user pilot WMS hides the identity of the users actually using the resources from the resource owners; the site administrators only see the pilot identity. This thus moves the trust relationship by the Grid sites from the final users to the WMS itself. It should however be noted that sites already rarely directly trust the final users, and that the trust is typically indirect by mediation through the VO. Since the WMS is typically run with the blessing of the VO, too, the implications of change in trust model are less severe than would otherwise be.

Nevertheless, many sites still want to know who is actually running on their resources, with the motivation ranging from simple desire of directly helping known users, to legal requirements. This requirement usually extends to the possibility of tracing every single operation by that user.

2.2. The virtual cluster security issues

The stated purpose of all multi-user pilot WMS's is to create virtual batch system clusters for their users. And any serious batch system is supposed to provide reliable protection between both jobs of different users, and to protect its own processes from the served users. In particular, a malicious user must not be able to access sensitive files of another user or the batch system itself, nor must he be able to send signals to processes of other users or the batch system itself.

Traditional batch systems normally achieve insulation between users by means of operation system protections, i.e. by running processes from different users under different identities, e.g. different UIDs under Linux. However, UID switching is only available to superusers, i.e. the Linux **root user.** Pilot job processes are however typically not running as root; doing so would require an exceptionally high trust from the resource owners.

Without the ability to perform UID switching, if a VO decides to deploy a multi-user pilot WMS, it must put a very high trust in its users; if a user were to turn up malicious, there is really no reliable protection in the system. As an example, one possible attack vector is for such user to impersonate the pilot WMS processes on the WN, by stealing the pilot credentials and using them to run code of his own choosing, by either injecting malicious code into the already running processes or by killing the existing ones first.

2.3. The role of glexec

In order to address the above issues, OSG and EGI have added glexec to the list of supported Grid middleware and have been encouraging member sites to deploy it on their resources.

Glexec is a tool that functions in a way very similar to the traditional Grid Gatekeepers, also known as Compute Elements (CEs), but is a privileged executable that can be invoked locally, instead of being a remotely invokable network service. Just like a CE, glexec receives a X.509 proxy certificate from the user, validates it, forwards the relevant information to the site's authorization and mapping service, and if all those steps succeeded, executes the user provided payload under the mapped UID. The way glexec obtains the proxy and the payload are of course different, but the functionality is comparable to that of a CE. A schematic view can be seen in Fig. 2.

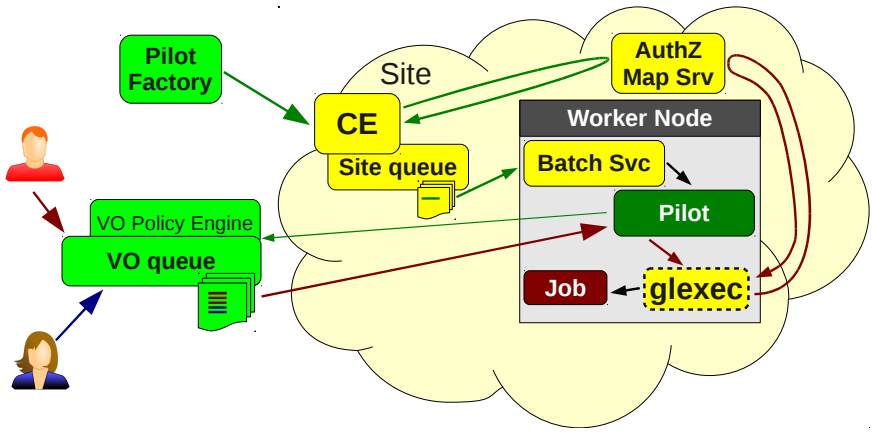

Figure 2. A schematic view of a glexec-enabled multi-user pilot WMS

When deployed on the site's worker nodes, and properly used by the pilot jobs, i.e. invoking it for each operation performed in the name of the actual user, glexec solves both of the issues mentioned in the previous subsections. By interfacing with the site's authorization system, the resource provider is given the identity of the actual user. Paired with UID switching, when each user identity is mapped to a different UID, this also allows the resource provider to associate any operation performed by any process on the node to the proper, global identity. And, finally, the pilot jobs can now perform UID switching and thus behave as a real batch system; please notice that UID switching from pilot to actual user UID is one way only, since the pilot has the user proxy, but the user does not have the pilot one.

However, it must be noted that pilot jobs cannot be forced to call glexec, and even when they do, there is no guarantee that the provided information is trustworthy; e.g. a pilot WMS could mix the credential of a user with the payload of a different user[1]. The site must thus still sufficiently trust the WMS owner, and any information obtained through the use of glexec must be considered as an enhancement and not as a replacement to the one obtained through the CE.

As a final note, glexec can also be deployed in non-privileged mode, i.e. configured to interface to the site's authorization system but not performing UID switching. Apart from not solving the pilot WMS's security problems, this operation mode provides no security benefits for the site either. Since in this deployment scenario glexec is not privileged, its functionality can be altered by any process running as the invoker UID, which includes both the pilot and user processes, and thus cannot be trusted.

3. glideinWMS integration with glexec

One multi-user pilot WMS implementation is glideinWMS and it builds on top of Condor to implement the pilot paradigm. The user facing services and the policy handling are implemented using the standard Condor services. The pilot itself is just a dynamically configured Condor resource-handling service, with glideinWMS providing the necessary logic to validate the resource and do the configuration. And, finally, there are proper glideinWMS services that perform site selection and submit the pilots. For the purpose of this paper, we will concentrate on the pilot part alone; the interested reader can get more information about glideinWMS and Condor in the referenced papers.

The glideinWMS pilot job workflow is composed of three steps. First, after fetching the needed files, the glideinWMS startup script runs a set of tests to validate the worker node it lands on. Using the gathered data, it then proceeds to create the needed Condor configuration files. And, finally, it starts the Condor resource-handling services, with the main Condor service being the **condor_startd**.

Integration with glexec is needed in two places. The main place is inside Condor services themselves, since they handle the launching, monitoring and termination of the user jobs. The glideinWMS startup script also needs to be aware of glexec, in order to properly configure Condor.

3.1. Condor integration with glexec

In every glideinWMS pilot, there are four Condor processes running. The main process is **condor_startd**, which has the task of managing the provisioned resource and negotiate with the rest of the Condor system which job to launch there. When a user job is received, a dedicated process named **condor_starter** is started, and this new process is then responsible of managing the user job for its whole lifetime. The reason for having two independent processes handling the provisioned resource is due to the fact that, if configured that way, the same pilot can run multiple jobs in parallel; e.g. if there is more than a single CPU available. Alongside those two, the third process is the **condor_procd**, who has the task of monitoring all the processes spawned by the other Condor daemons, and send them

[1]Just for completeness, it should be noted that the same problem exists with the CEs, too, any time a user delegates his credential to a resource broker.

signals, if needed. Finally, there is the **condor_master** who has the task of starting and stopping all the other Condor processes.

Condor supports three modes of operation for job execution:

1. Condor runs as root and executes jobs as other users
2. Condor runs as non-root and executes jobs as the same user
3. Condor runs as non-root and uses a privileged tool to execute jobs as other users

The first mode is typically used by the Condor batch system that is installed and managed by the system administrator, and is not an option for a pilot WMS. The second option is what is used in glideinWMS pilot jobs when glexec is not available. This section describes the third option. Please also notice that Condor supports UID switching through privileged tools other than glexec, but that part will not be described in this document.

Integration with glexec is needed in two places; in the condor_starter and in the condor_procd. The condor_starter is responsible for launching, monitoring, and cleaning up user jobs and associated files. It invokes glexec for several purposes; to set up the input files for the job, launch the job, periodically update the job's proxy certificate, and to change the ownership of the job output files back to the pilot user. In addition, when Condor needs to kill processes associated with the job in various situations, it delegates the task to condor_procd daemon, which in turn invokes glexec.

There are two interfaces between Condor and glexec. For most commands, e.g. sending a signal and running **rm**, Condor simply invokes glexec by providing the user proxy certificate and passing as payload the UNIX command to execute. For starting up the user job, the Condor team has instead had to come up with a sophisticated wrapper, named **condor_glexec_job_wrapper**; this was needed in order to properly handle the edge cases.

The simple invocation of glexec can nevertheless be quite powerful. For example, setting up the job's working directory is accomplished by piping the output of **tar**, who reads files owned by the pilot user, into the input of a tar invoked by glexec, and who thus writes files using the final user idenitity, as in:

- `(cd $DIR && tar -cf – subdir) | glexec tar -xf -`

However, when condor_starter needs to launch the user job, it passes condor_glexec_job_wrapper as the payload to glexec. The job wrapper communicates with condor_starter over a pipe that it receives as its standard input. The job's desired environment and standard input, output, and error descriptors are passed over the pipe. The job wrapper then executes the job or communicates back a descriptive error in case of failure. The communication channel between the job wrapper and the condor_starter has the close-on-exec flag set so the starter can expect to either read an error message on this pipe or an end-of-file in case of a successful call to exec() when launching the job.

Finally, although already implied above, we would like to stress that Condor calls glexec several times during the lifetime of a job; at a minimum, it will call it once to set up the working directory, once to start the job and once to retrieve the outputs. Additional calls may be needed for the re-delegation of an updated proxy and/or for sending signals to the job processes.

3.2. glideinWMS startup script integration with glexec

The glideinWMS pilot job startup script must be aware of glexec in order to properly configure Condor. There are three steps in the integration process.

First, the startup script reads its own configuration to determine if glexec should be used or not; there are three possible states:

- glexec use is **required**, and the pilot job must fail validation if it cannot be used
- glexec use is **optional**, i.e. use glexec if available, continue without it else
- **never** use glexec, even if available, i.e. do not even check if glexec is installed

The reason for the different options is due to the different needs of the various deployers of the multi-user WMS. On one end of the spectrum, some VOs may be very security conscious and prefer to restrict themselves to glexec-enabled Grid sites only; on the other end, some VOs may have users without X.509 proxy certificates, and thus cannot make use of glexec. And in between we have the pragmatic VOs who will take advantage of glexec when possible, but will live with the security risk on the other Grid sites.

The second step is validating the glexec installation, when it is to be used. The test consists of locating glexec, invoking it with the pilot X.509 pilot proxy and validating the result. Using the pilot proxy is of course not ideal, since it does not tell us if the actual final user will be authorized or not; however, it is the only credential we have at this point of execution, and it does catch the obviously broken glexec installs. Two tests are currently used:

- `glexec sh -c "id && echo \"Hello World\""`, and
- `glexec "$PWD/glexec_test2.sh"`

The two independent tests are needed to catch different types of misconfiguration that we noticed in the few years of running. The first one can fail due to bugs in wrappers put in place by system administrators. The second one can fail due to improper directory permissions in the directory tree. Of course, if core glexec is improperly configured, both will fail, too.

The last step in the startup script is to propagate the discovered values to Condor. In essence, if glexec is to be used, the script tells Condor where to find glexec and requires its use.

4. The evolution of the glideinWMS integration with glexec

The integration of glideinWMS with glexec has evolved during the years. As we gained operational experience, and expanded our user and resource provider base, several problems have been found that required changes in our software stack. While the majority of them were merely annoyances, some were posing a security risk, while a precious few were actual show-stoppers.

This section presents the problems that were found and how they were fixed. The critical problems are listed first, followed by the security risks and then all the other ones. Within each category, the problems are listed in chronological order.

4.1. Condor discovered not working with glexec in linger mode

Glexec has two operation modes: after authentication and authorization, it can either change UID and replace its own image with the target executable, or fork a new process under the proper UID for the target command. This second mode of operation is called **linger mode**.

Initially, the communication between the condor_starter and condor_glexec_job_wrapper was relying on the later closing its standard output handle to signal success. This condition is however not propagated to the caller of glexec in linger mode, because glexec retains a handle to the pipe. The result was that condor_starter would hang for the duration of the job and would therefore not provide runtime monitoring and job proxy updates.

The solution was for the condor_starter and condor_glexec_job_wrapper to share a pipe directly rather than relying on inheritance through glexec. This is accomplished by passing the pipe file descriptor over the wrapper's standard input. This now works as expected for both glexec operation modes.

The final fix required a new version of Condor binaries to be used by the glideinWMS pilots.

4.2. Condor discovered not working with the EGI version of glexec

Due to various historical reasons, OSG started supporting production glexec deployments several years before EGI (and its predecessor, EGEE) did. Condor integration with glexec was thus developed and tested with the version of glexec deployed on OSG. Once the first deployments of glexec were available at a few EGI sites, we tried to make use of it, just to discover that Condor would not work at all with the version of glexec deployed there.

The root cause of the problem was an evolution in semantics of glexec, paired with the difference of site authorization systems used by the two Grid infrastructures. In OSG, only the final user credential is used to authorize the invocation of glexec; on EGI, both the pilot and final user credential are needed. Condor was thus passing only the final user proxy to glexec, which worked just fine on OSG, but resulted in permission denied on EGI.

The symptoms were also quite weird; the simple invocation of glexec was working fine, but invocation of condor_glexec_job_wrapper was failing. The reason for this disparity was the way Condor was invoking glexec to execute condor_glexec_job_wrapper. The pilot proxy is being passed to glexec through an environment variable, in particular X509_USER_PROXY, and this happens to be properly set to the pilot proxy by the glidein startup script while starting up Condor. However, when invoking glexec to execute condor_glexec_job_wrapper, Condor would set that variable to the user proxy; while it would instead leave it alone for simple glexec invocations.

The setting of that environment variable when starting condor_glexec_job_wrapper turned out to be a logical bug in Condor, which happened to be completely harmless in the OSG use case, and the fix was simply using a similar logic for both glexec-invoking code paths.

The final fix required a new version of Condor binaries to be used by the glideinWMS pilots.

4.3. Directory tree permission problems

One property of glexec is that it will change the UID between the time it is called and the time the target executable is launched. A side effect of this property is that the target binary must be accessible and readable by both the calling user and the target user. Which, in turn, has the side effect of requiring that also the whole path leading to the target binary must be searchable by both users.

Most of the time, this is not a problem in batch environments; the work directories are typically world searchable. However, when the work directory happens to be in a sub-directory of the user home directory, it is likely that the target user does not have access to the work directory anymore. This was indeed the case for more than one EGI site we used for the initial round of glexec-on-EGI tests; resulting in Condor not being able to use glexec.

The solution advocated by the glexec development team, and approved by both EGI and OSG security teams, is to relax the permissions of the whole path leading to the work area, making each and every directory in the path world searchable. This task was implemented in the glidein startup script.

The final fix required a new version of glideinWMS to be deployed by glideinWMS operators.

4.4. Protecting the condor_starter from the user

The first integration of Condor with glexec used a slightly different architecture from the one described in Section 3. In order to minimize code development, the condor_startd would execute the condor_starter via glexec; this was deemed easier to implement due to the clear interface between the two processes.

The condor_starter, however, has a privileged relationship with the other Condor daemons and is also responsible to monitor and manage the user job. Thus running it under the same UID as the user job is a security risk, since the user could attack it. Recent versions of Condor thus keep the condor_starter running with the pilot UID.

The final fix to this problem required both a new version of the Condor binaries to be used by the glideinWMS pilots, as well as changes to the glidein startup script to support this new feature.

4.5. Losing control of the user job

If Condor uses glexec to start a user job, it must use glexec for all the following operations as well, including monitoring the job, retrieving the output sandbox, and killing the job, if needed. There is however no guarantee that Condor will be able to do it; each and every invocation of glexec is authenticated and authorized independently.

There are some situations where Condor is denied the use of glexec for reasons beyond its control; e.g. if the sites decides to ban a user or if the site's authorization service dies. There are however a few other situations where Condor needlessly loses control of the job, with the two major being user proxy expiration and change in user proxy identity.

Condor owns the user proxy and can inspect it at will; it could thus easily determine when the proxy is about to expire, and thus when glexec will stop accepting requests in the user's name. Older versions of condor_starter do not check for proxy lifetime, and thus let the proxy expire without taking any action. Newer versions of condor_starter will instead kill the job and clean after it a few minutes before the proxy expiration.

Condor also allows users to re-delegate their proxy certificates, in order to allow them to keep them as short lived as possible. The re-delegation is supposed to be a renewal of the same X.509 credential, with the same extended attributes, if any. There is, however, currently no strict requirement that the identity of the new proxy is the same as the old proxy; and if the user were to re-delegate a proxy with a different identity, the target UID after the glexec invocation will likely not be the same either. There is currently no protection in Condor for this situation, and the condor_starter will simply overwrite the file containing the user proxy and thus lose control over the job it started. Recent versions of Condor will at least register the change in identity, but a complete solution to this is currently still in the works.

Finally, Condor also does not properly handle transient glexec problems; if a call to glexec returns with authentication denied, Condor assumes that this situation is permanent, and will give up on a running job. Unfortunately, glexec does not provide a way to distinguish temporary errors from permanent ones, e.g. authorization service overloaded vs user credential revoked; nevertheless, Condor should be smarter and at least retry a few times. A solution to this is also still in the works.

4.6. Avoiding glideins being a DoS vehicle

Each invocation of glexec puts a non-negligible load on the site's authorization and mapping service. Given that a typical site has O(1k) Grid jobs running at any point in time, this can rapidly add up, if all of them were to call glexec at exactly the same time. And there was a situation when glideinWMS would synchronize the glexec invocations, resulting in a Denial of Service (DoS) attack to the site services.

Condor tries to be as efficient as it can be; so, if a user were to ask it to cancel a large number of computing jobs, it would do it as fast as it could. From the job queue point of view, canceling a running jobs effectively means sending a kill command to each and every condor_starter handling a job; a very lightweight and fast operation. The various condor_starters would thus receive the command within milliseconds from each other, and would all attempt to execute it immediately.

To avoid this situation, Condor has added the possibility to configure the job queue to spread the sending of commands over a longer period of time, thus reducing the impact on the site. This has however the side effect of wasting CPU at target sites, since jobs that are known to be useless are kept running, but this is a price that has to be paid.

Please also notice that this solution is completely in the hands of the WMS operator; the sites are still vulnerable. To the best of our knowledge, there is currently no fix for that, but this is beyond the scope of this document; Condor, part of the glideinWMS software stack, does the best it can.

4.7. Smarter handling of authorization denial

One reason Grid sites deploy glexec is to retain the control over who uses their resources. Which implies that they can deny access to any user they elect to; i.e. an invocation of glexec for certain user credentials will always fail. The condor_starter of course respects this decision, and will not start a user job unless the call to glexec succeeds; however, it will not blacklist that user.

When condor_starter cannot start a job, it notifies its parent, the condor_startd, of this fact, and the condor_startd releases the current claim and re-negotiates the CPU with the rest of the Condor system. The negotiation procedure will look for the job with the highest priority and assign it to the requesting condor_startd. Unfortunately, it is very likely that the highest priority job is still the job that was just refused, getting the glidein into a useless try-and-fail loop. This can also result in a DoS attack against the site's authorization system.

In order to address this problem, recent versions of Condor put the denied job into held state, alongside a descriptive error message, effectively preventing it for being re-matched. While this is not ideal, since the job may have been able to start at a different Grid site, it was deemed the most effective solution. If the glideinWMS administrator wants to implement a smarter policy, it can do it by putting in place a mechanism that changes the job requirements followed by a resumption of the job.

4.8. Improved glexec validation

Several sites have customized their glexec deployments, by adding custom wrapper scripts, resulting in errors when Condor tried to use glexec, while still passing the initial validation test. The validation test has thus been tweaked several times to catch as many known failure modes as possible.

4.9. Minor Condor bugs

There were two further, minor glexec-related Condor bugs worth mentioning.

The first bug resulted in condor_glexec_job_wrapper setting the PATH environment variable passed to the user job to the one inherited from glexec, instead the one specified in the Condor configuration file.

The second bug resulted in condor_ssh_to_job, an interactive debugging tool, to not work when glexec was used by the condor_starter. This was due to the requirement in the protocol to specify the used UID, and the client was passing the pilot instead of the final user one.

Both are fixed in recent versions of Condor.

5. Current use of glexec by OSG glideinWMS instances

The glideinWMS architecture is composed of two distinct components; a VO specific component, usually referred to as a VO frontend, and a shared service, usually referred to as a glidein factory. OSG is operating one glidein factory instance[6,7] for many OSG-affiliated VOs, spanning biology, chemistry, climate science, computer science, economics, engineering, mathematics, medicine and physics. This allows us to provide information on how these VOs use glideinWMS, in particular in relation to glexec.

The OSG glidein factory currently sends glideins to 297 CEs, distributed over 172 sites[2] worldwide. 55 of these use the OSG stack, and are located in the Americas, while the rest is based on EGI provided software, spanning the other continents. Glexec has been tested and enabled for glideins at 49 CEs, which are distributed over 26 sites. 13 of those are based on OSG and the other 13 are based on EGI provided software stack. Only one (1) site requires the use of glexec for all multi-user pilot WMS's.

[2]In the context of this section, a site is identified by the DNS domain name.

The OSG glidein factory serves 15 VO frontends, 13 of which can be classified as multi-user pilot WMS's. None is requiring the use of glexec, but 9 will use it when available. On top of that, 2 VO frontends have special rules in their VO frontends that will enable glexec only on the (lone) site requiring the use of glexec.

The above numbers are summarized in Table 1.

Table 1. Resources used by the OSG glidein factory instance

	Total	**glexec-enabled**
CEs	297	49
Sites (OSG sites)	178 (46)	23 (9)
Sites requiring the use of glexec	1	1
VO frontends	15-2	9+2

The operational experience with operating glexec-enabled glideins has generally been very positive. Glexec-specific validation errors are typically pretty negligible; as an example, in the week of May 14th, glexec validation tests for a major VO frontend failed on less than 200 glideins, out of a total of 30k, i.e. a fraction of a percent. When there are problems, they are typically due to a broken WN installation, or an overload of a site authorization and mapping service.

Nevertheless, enabling glexec on a new site was often challenging. This was particularly true in the years past, when sending glideins through our glidein factory was the only reliable way to test a glexec installation. Recently, the physics community started validating their sites with standalone tests, which mimic the glexec validation tests provided by glideinWMS, so enabling glexec on more sites will hopefully be less time consuming.

6. Conclusions

Many VOs have adopted the multi-user pilot-based WMS paradigm and this brought with it a change in the security model, since a single, pilot credential is used to provision resources that will be used by jobs from many physical users. In order to both propagate the final user identity to the resource providers and to preserve OS-level protections between different users, OSG and EGI have deployed a privileged executable on the worker nodes, called glexec, to be used by the pilot WMS's.

A broadly adopted multi-user pilot WMS implementation is glideinWMS, which heavily relies on Condor to implement the pilot paradigm. Condor, and by extension glideinWMS, had support for glexec essentially since the day the tool was released, and a few OSG VOs have been using glexec for this whole time, with several others joining later on. In this time span, we have discovered several problems in the integration of Condor with glexec, most of which have been addressed shortly after being discovered. The two issues that still need work are a smarter way of dealing with changes in user proxy identity after a job start, and smarter handling of transient glexec failures.

At the time of writing, about 85% of OSG VOs using glideinWMS have enabled the use of glexec in their instances. The support from the sites is instead much lower, with glexec being installed and tested on only about 15% of the Grid sites used by the OSG glideinWMS instances. As a consequence, no VO is running in a glexec-only mode yet. Nevertheless, glexec is now being routinely used where available, and the operational experience has generally been very positive, with glexec-specific errors being typically pretty negligible.

Acknowledgements

This work is partially sponsored by the US National Science Foundation under Grants No. PHY-0612805 (CMS Maintenance & Operations), OCI-0437810 and OCI-0850745 (Flight Worthy Condor) and the US Department of Energy under Grants No. DE-FC02-06ER41436 subcontract No. 647F290 (OSG), and No. DE-AC0s-07CH11359 (Fermilab).

References

[1] Pordes I et al. 2007 The open science grid *J. Phys. Conf. Ser.* 78, 012057. doi:10.1088/1742-6596/78/1/012057.

[2] Kranzlmüller D, Marco de Lucas J and Öster P 2010 The European Grid Initiative (EGI) Towards a Sustainable Grid Infrastructure *Remote Instrumentation and Virtual Laboratories* Part 2, 61-66, doi:10.1007/978-1-4419-5597-5_6.

[3] Groep D, Koeroo O, Venekamp G 2008 gLExec: gluing grid computing to the Unix world *J. Phys.: Conference Series* 119 062032. doi:10.1088/1742-6596/119/6/062032.

[4] Sfiligoi I, Bradley D C, Holzman B, Mhashilkar P, Padhi S and Würthwein F 2009 The Pilot Way to Grid Resources Using glideinWMS *Computer Science and Information Engineering, 2009 WRI World Congress on*, vol. 2, pp. 428-432. doi:10.1109/CSIE.2009.950.

[5] Thain D, Tannenbaum T and Livny M 2005 Distributed computing in practice: the Condor experience. *Concurrency and Computation: Practice and Experience*, vol. 17, no. 2-4, pp. 323-356. doi:10.1002/cpe.938.

[6] Sfiligoi I et al. 2011 Operating a production pilot factory serving several scientific domains *J. Phys.: Conf. Ser.* 331 072031. doi:10.1088/1742-6596/331/7/072031.

[7] Sfiligoi I, Würthwein F, Dost J M, MacNeill I, Holzman B and Mhashilkar P 2011 Reducing the Human Cost of Grid Computing With glideinWMS *Proc. of CLOUD COMPUTING 2011, The Second International Conference on Cloud Computing, GRIDs, and Virtualization* pp. 217-221. ISBN:978-1-61208-153-3.