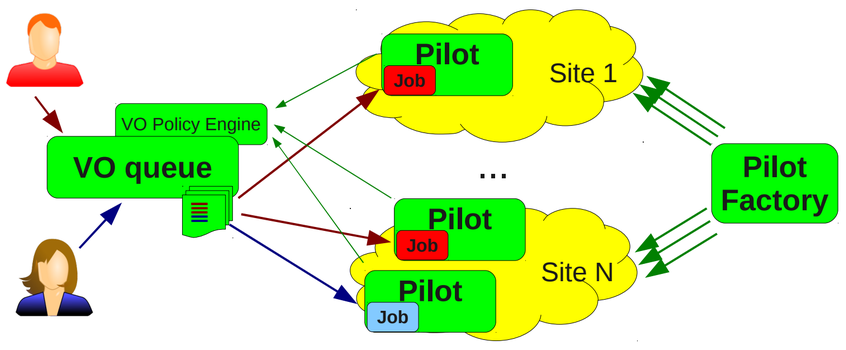
glideinWMS experience with gLExec

by
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

Many VOs have adopted the **pilot-based WMS paradigm**. In this paradigm, resources across multiple administrative domains are aggregated into VO-specific overlay pools by means of pilot jobs.

Each VO has full control over its own pool, 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.



Deviates from Grid security model. The pilot credential not specific to the user of which the job is ran. Thus called **multi-user pilot jobs**.

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.

Naïve use of multi-user pilot WMS has major security implications.

By submitting multi-user pilot jobs, the multi-user pilot WMS's **hide 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.

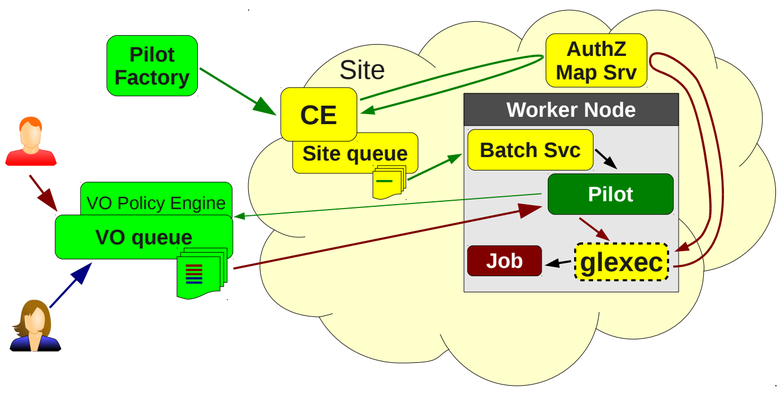
Pilot job processes are typically not allowed to run as root, thus cannot perform UID switching. Without this functionality, they **cannot use OS-level insulation** from the users. If a VO decides to use a multi-user pilot WMS, must have very high trust in users.

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. 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.

OSG and EGI deploying glexec to provide a solution.

Glexec is a tool that functions in a way **very similar to the traditional 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.



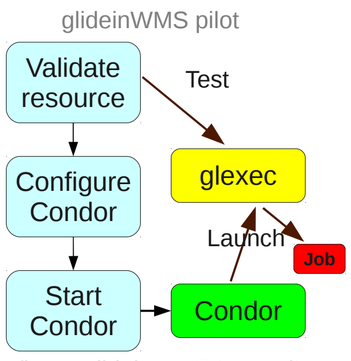
UID switching from pilot to actual user UID is one way only, since the pilot has the user proxy, but the user does not have the pilot one.

When deployed on the site's worker nodes and properly used by the pilot jobs, **glexec solves both issues**.
The site admin gets the user credential and the pilot can perform UID switching.

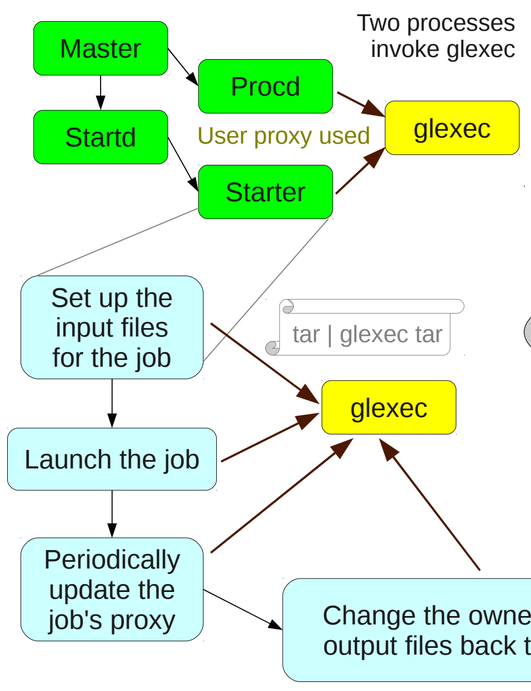
By interfacing with the site's authorization system, the resource provider is given the identity of the actual user. Paired with UID switching this also allows the resource provider to associate any operation performed by any process on the node to the proper, global identity. And, finally, the pilot jobs can now perform UID switching and thus behave as a real batch system.

glideinWMS has been integrated with glexec

glideinWMS builds on top of Condor to implement the pilot paradigm. The pilot job is a **dynamically configured Condor** service, with a wrapper validating the WN and doing the configuration.



The use of glexec is optional. The VO can configure glideinWMS to use it or not.




```
condor_procd
# Main use is to kill processes
glexec /bin/kill $PID
```

```
Launch the job
# Needs a dedicated wrapper
glexec $SBIN/condor_glexec_job_wrapper
```

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 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.

Two tests ran in the wrapper:

```
# Apart from testing for core glexec problems
# Make sure no buggy wrappers in place
glexec sh -c "id && echo \"Hello World\""
# and that the directory tree is kosher
glexec "$PWD/glexec_test2.sh"
```

Pilot proxy used to test glexec. (Not ideal but the only available.)

Not a single step ... it is a journey

Sometimes we hit show-stoppers

glexec in **linger** mode

Old versions of Condor relying on the wrapper 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.

OSG vs EGI glexec

Condor integration developed on OSG.
In OSG, only the final user credential is used to authorize the invocation of glexec; in EGI, both the pilot and final user credential are needed. Condor was only passing the user one.

Directory permissions

Whole path leading to the target binary must be searchable by both caller and target user.
Now fixing permissions during validation.

Then there are residual risks

condor_starter after glexec

In order to minimize code development, initially the condor_startd would execute the condor_starter via glexec. But condor_starter is privileged, Thus running it under the same UID as the user job was a security risk.

Loosing control over job

Cannot glexec after job start

Proxy expires

Condor was not looking at proxy lifetime.
Now will kill the job and clean after it a few minutes before the proxy expiration.

Proxy changes

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.

No solution yet.

Transient problems

Condor treating all glexec errors as permanent.
Need smarter handling of transient errors.

Improvements due to experience

Glideins as DoS vehicle

If O(1k) jobs canceled, authorization storm from O(1k) glexec calls.
Now can spread job cancellation.

Try-and-fail loop

Site can blacklist a user.
If glexec fails, Condor will rematch the glidein.
But best job likely the one who failed.
Now put failed job on hold.

Better glexec validation

Several Condor bugs

Current statistics

Resources supported

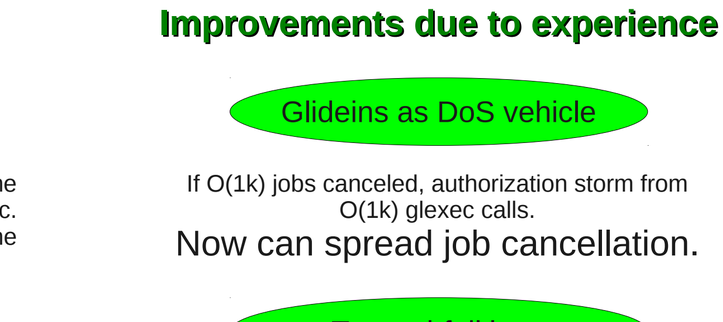
	Total	glexec-enabled
CEs	297	49
Sites (OSG sites)	178 (46)	23 (9)
Sites requiring the use of glexec	1	1

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 for many OSG-affiliated VOs and the data relate to this instance.

How VO frontends use glexec



The **operational experience** with operating glexec-enabled glideins has generally been **very positive**. Glexec-specific validation errors are typically pretty negligible; e.g. 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, WLCG started validating the sites, which mimic glexec tests provided by glideinWMS, so enabling glexec on more sites will hopefully be less time consuming.

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).