OSG glideinWMS Factory operations input on Grid middleware

I. Sfiligoi and J. Dost, UCSD

Executive summary

The glideinWMS Factory (GF) instance operated by OSG[1] is today the only interface to the Grid for most of the OSG VOs. Over the past few years, the OSG GF operations team has accumulated extensive experience with the Grid interfaces use to submit jobs both to OSG and the partner Grid sites, e.g. EGI. In the process we formed the opinion that the abstraction layer introduced by the "Compute Element" software provides no real value to us, but instead introduces significant problems. We furthermore believe that this is a fundamental problem due to the existence of the abstraction layer, and not just a software problem of a particular implementation.

We thus advocate that the OSG software stack should move its operational model from a "hide as much as possible" to "provide as low level access as possible". One possible technical solution that would be a step in that direction is GSI-authenticated SSH.

OSG Grid Architecture Description

Virtually all OSG users are today using the pilot paradigm to use resources on OSG. In this paradigm, a VO creates the overlay scheduling infrastructure that dynamically provisions the resources for their users, and the final users themselves only ever see this overlay. The overlay scheduling infrastructure is composed of two logical parts:

1. The VO hosted jobs queue(s) and matchmaking mechanisms
2. The resource managing processes running on OSG resources, i.e. the pilots

In order to get the pilots to the actual OSG resources, i.e. the worker nodes (Wns), the VO's provisioning layer submits the pilots as compute jobs to the OSG-provided interface at the various sites; this interface is usually called the Compute Element (CE). The CE converts the request into a submission to the local scheduling system with minor site-specific customizations. The CE is also responsible for providing monitoring information about the pilot job, again, as a proxy against the site-local scheduling system.

Once the pilot job starts, it connects back to the VO hosted infrastructure and generally operates independently of the site-level Grid infrastructure.

The glideinWMS is the pilot provisioning system used by the majority of OSG VOs, with the VO-specific scheduling layer being implemented as vanilla HTCondor. In glideinWMS, the provisioning functionality is split it two independent pieces; the VO Frontend (FE) implements the provisioning logic while the Glidein Factory (GF) is the actual interface to the Grid. As a consequence, only the GF interacts with the Ces. The FE and the GF can, and often are, operated by independent entities, with a N-to-M relationship between them. OSG operates[1] three GF instances with a single operational team, and these GFs are being used by the majority of OSG VOs for all their Grid needs.

1 Jointly with CMS, but as a single team lead by the authors of this paper who are paid by OSG

Questioning the value of the current Grid infrastructure

The current CE paradigm was drafted with the idea of having a single interface to any number of site-local scheduling systems. This interface was supposed to solve thee problems:

1. Simplify the access logic, by providing a single API.
2. Provide a mean to throttle jobs submission, if needed.
3. Provide a mean to customize job submission for the sites.

Today, however, none of the above apply. When taking into account the partner Grids, we have to deal with four different CE technologies, which is comparable in number to the number of site-local scheduling systems. The main OSG CE implementation, namely Globus GRAM, does not provide any useful throttling. And the site customizations are not really transparent to the users, requiring site-specific tunning for most jobs.

From the user point of view, thus, there is really no benefit to the current infrastructure; the authors can authoritatively state this, since we are responsible for a very large fraction of Grid job submissions. But there are several drawbacks, as explained in the next section.

For completeness, we now try to look at the site benefits; the authors acknowledge that they cannot make an authoritative statement on this, but they still have a significant experience in the field. Being able to customize the jobs before they are submitted to the local batch system is likely the major, and possibly the only benefit a site gets from the current CE architecture. This is however only needed at submission time, so adding a transformation layer also for monitoring purposes does not help.

Theoretically, the sites would also benefit from throttling mechanisms, if they were available. However, with the advent of the pilot paradigm, the number of users each site sees has been dramatically lowered and these remaining users are expected to be well behaved as well, so the benefit would really be minor. Today, each OSG site has to deal with only a handful of job submitters; the OSG glideinWMS factory operators plus a few non-GF users.

Actual problems with the current Grid infrastructure

The glideinWMS Factory operations effort is currently dominated by shortcomings introduced by the current Grid infrastructure operational model. The problems are of two kinds; problems due to lack of information and problems with the software implementation itself.

The glideinWMS Factory operations main task is to detect problems with WNs at the sites of various resource providers, and to steer the administrators of those resource providers toward a fast resolution of those problems. It is our experience that most sites do not have proper monitoring in place, so the majority of them have at least some kind of problem at any point in time.

The abstraction layer of the CE makes our task relatively hard for most non-trivial problems. The only mechanism we have to understand what happens to the pilots in the early stages of their lifetimes, i.e. before they can talk to the VO-hosted services, are the logs they ship back at termination. However, when we hit black hole nodes, no logs are ever returned. This makes it both difficult to diagnose the problem, and once diagnosed, provide useful information to the site administrators for prompt resolution; several email round trips are often needed. If we could get direct access to the site scheduling layer and its job logs, we would probably be able to cut our effort and time-to-resolution by at least an order of magnitude.

An even more tricky problem is when the site refuses to accept a job submission request. The error message obtained is often very cryptic and hard to act upon. Having a clear separation between access to the site's scheduling system and actual submission to the scheduling system would likely significantly clear the situation, and thus save significant effort now wasted in trial and error attempts.

Finally, there are software problems, the most problematic currently being Globus GRAM forgetting about the submitted jobs. This results in the GF improperly counting the pilots being provisioned and thus not provisioning the amount requested by the VO. This could arguably be mitigated by smarter glideinWMS software logic, but at this point in time it does account for significnat effort of the GF operational team.

More details can be found in the related paper, that provides the GF Ops perspective on the Grid.

Proposed solution – Direct access to the site scheduler

So, we here argue that OSG should change the operation model, and provide only a lightweight interface in front of the site's scheduling system. This interface should deal with GSI authentication and job submission customization only.

The best technical solution we are aware of is GSI-enabled SSH, with some kind of lightweight wrappers around the job submission commands. Both techniques are currently being used with success in OSG. The GSI-SSH is being used by CMS for their own user portal. While lightweight wrappers are successfully being used by OSG Connect.

As users, we GF operators believe it will make our lives as Grid users much easier. The overhead of having to learn to deal with different scheduling systems at sites will be more than offset by reduced ongoing operational effort due to access to lower level information.

We also believe that sites will find this solution much more palatable, since they will not have to learn to use and operate an “exotic Grid service”, but will have to operate a standard SSH service instead, like they probably already do for the other users of their resources.