THE glideinWMS APPROACH TO THE OWNERSHIP OF SYSTEM IMAGES IN THE CLOUD WORLD

Igor Sfiligoi[1], Anthony Tiradani[2], Burt Holzman[2] and Daniel C. Bradley[3]
[1]*University of California San Diego, 9500 Gilman Dr., La Jolla, CA 92093, U.S.A.*
[2]*Fermi National Accelerator Laboratory, Wilson and Kirk Roads, Batavia, IL 60510, U.S.A.*
[3]*University of Wisconsin–Madison, 1210 W. Dayton St., Madison, WI 53706, U.S.A.*

Keywords: Image Ownership, Cloud, glideinWMS.

Abstract: Scientific communities that are accustomed to use Grid resources are now considering the use of Cloud resources. However, moving from the Grid to the Cloud brings along the need for the creation and maintenance of the system image used to configure the provisioned resources, and this presents both opportunities and problems for the users. The impact is especially interesting in the context of glideinWMS due to its layered architecture. This paper describes the various options available to the glideinWMS project team, their advantages and disadvantages, and explains why one of them is to be preferred.

1 INTRODUCTION

Scientific communities that are accustomed to use Grid resources are now considering the use of Cloud resources (Mell and Grance, 2011). In this context, Cloud is equated with the Infrastructure as a Service (IaaS) paradigm.

One major difference of the Cloud model compared to the Grid model is the users' ownership of the Operating System (OS) and related libraries, both regarding installation and configuration. This paper analyses the impact of this new capability in the context of the glideinWMS project (Sfiligoi et al., 2009).

Section 2 provides a rationale of why users like the Cloud model, section 3 provides a description of the glideinWMS system, while section 4 provides the analysis of the various options available to the glideinWMS project. Finally, section 5 provides an overview of the expected future work, followed by a conclusions section.

2 WHY CLOUDS

There are several reasons why Clouds are being considered by our **user communities, known also as Virtual Organizations (VOs)**. In this paper, however, we limit ourselves only to one specific aspect of the Cloud ecosystem; the ability, and requirement, of instantiating a complete compute environment.

In the Grid model, users are allowed to start a process of their choosing, but the system services and libraries are provided by the resource provider. This implies that the user can only use the available resources, if he can adapt his code to work in the provided environment. The Cloud model instead calls for the provisioning of typically virtualized low level hardware. The user thus provides a whole **system image** to be started, including both the system services and libraries, and his actual computational code. The perceived benefits are two-fold; on one side, the user can perform system level operations, which were impossible in the Grid model, and on the other he gets access to a more homogeneous set of resources.

The homogeneity claim needs some clarification. In theory, the variety of hardware being deployed is one of the main reasons why users need an OS as an abstraction; requiring the users to provide the OS thus seems backwards. However, often times it is easier for a user to properly configure an existing operating system on arbitrary hardware than modify his scientific code to work on a drastically different OS. Moreover, thanks to the virtualization layer, most Cloud providers do provide only a very limited number of variations of virtualized hardware.

Sfiligoi I., Tiradani A., Holzman B. and Bradley D. (2012).
THE glideinWMS APPROACH TO THE OWNERSHIP OF SYSTEM IMAGES IN THE CLOUD WORLD.
In *Proceedings of the 2nd International Conference on Cloud Computing and Services Science*, pages 443-447
DOI: 10.5220/0003961904430447

3 CURRENT USES OF glideinWMS

The **glideinWMS** project is currently being used to manage job scheduling on world-wide Grid resources by O(10) scientific communities. Some of the authors have also been intimately involved for several years in the operations of a significant fraction of the glideinWMS-related deployed infrastructure; we thus have a significant insight into the desires and needs of the user communities we serve.

In this section, we first provide a summary description of glideinWMS and then analyse how our user communities use it.

3.1 The glideinWMS Architecture

The glideinWMS is a pilot-based Workload Management System (WMS), creating an overlay batch system on top of provisioned Grid resources. One important aspect of glideinWMS architecture is the existence of three clearly separated layers, as depicted in Figure 1:

- a Grid interfacing layer, called the **Glidein Factory**;
- a resource provisioning logic layer, called the **VO Frontend**; and
- the **proper, overlay WMS**, which is seen by the users.

This clear separation allows for factoring out of the operations to separate administrative groups, allowing for both increased operational efficiency due to specialization, and lower operational overhead at large scale due to minimized interactions between the various smaller groups. Moreover, a nice side benefit of this architecture is that the Glidein Factory is completely generic, and can serve multiple VO Frontends, further reducing the operational costs (Sfiligoi et al., 2011b). In a similar manner, a VO Frontend can be served by multiple Glidein Factories, thus eliminating the factory component as a single point of failure. Similarly, a WMS can be provisioned by multiple VO Frontends. See Figure 2 for an overview.

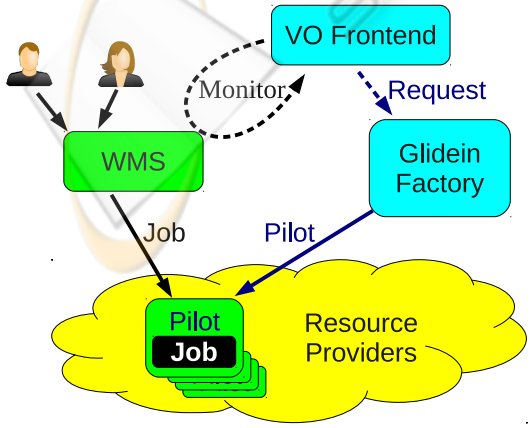

Figure 1: The glideinWMS architecture.

Since the system is N-to-M in nature, it is important that **each logical element has a very well defined role** in the provisioning process; this way, instances can be added or replaced without significant changes observed by the users of the WMS. In glideinWMS:

- the WMS is responsible for maintaining the proper security infrastructure for its central service and its users, as well as handling the prioritization policies;
- the VO Frontend is responsible for managing the credentials used to provision the resources, providing or validating the availability of tools and libraries used by the served user community, and defining the matchmaking policies; and
- the Glidein Factory is responsible for interfacing with the resource providers, doing the system-level validation of the provisioned resource, providing binaries needed by the pilot, and properly configuring them based on the type of resource acquired.

3.2 glideinWMS in Practice

Currently, the user communities are split in two distinct categories; those who decided to operate all the pieces of the glideinWMS system themselves, and those who decided to offload the operations of Glidein Factories to external groups. To the first approximation, nobody split the operations of a WMS from the operations of a VO Frontend, although we are aware of a user community where there is only partial overlap between the two groups.

The communities operating the full set of services tend to be experienced Grid users, who started

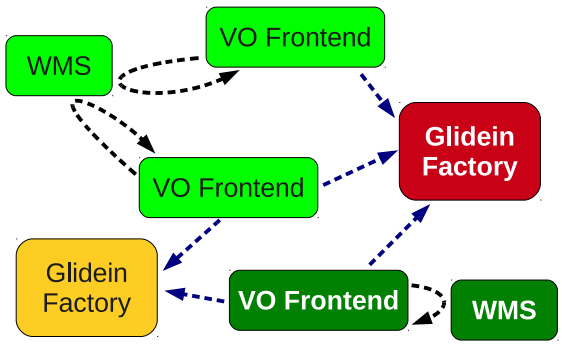

Figure 2: The N-to-M cardinality of the glideinWMS components.

with direct Grid submission and are unwilling to completely break the ties. While important players, they are a shrinking minority, so they will not be further considered in this paper.

The remaining communities are served by only a couple Glidein Factories, each of which covers most of the available worldwide Grid resources used by these communities. The funds for operations of these Grid Factories is provided by large-scale Grid infrastructures, the most prominent being the Open Science Grid (OSG) (Sfiligoi et al., 2011a). Please notice that several of the authors are involved with operations of either a VO Frontend+WMS instance or a Glidein Factory instance.

In our experience, the **operational tasks of the Glidein Factory and the VO Frontend+WMS operators are clearly distinct**:

- the Glidein Factory operators are responsible for monitoring and reacting to the changes in the Grid resource configurations, discovering and troubleshooting misbehaving resources, and for keeping up to date with updates in the pilot software; while
- the VO Frontend+WMS operators are responsible for keeping up to date with the needs of the users, thus modifying accordingly both the validation scripts and the matchmaking expressions, and for keeping up to date with the updates in the central services software.

The VO Frontend+WMS instances are extremely varied, and include small departmental groups, campus-wide deployments of mostly independent scientists, and portal-based, well-organized international scientific collaborations. Their configurations and operational procedures are understandably just as varied, so they will not be described in this paper. Nevertheless, they all report very low operational costs, with the vast majority of effort being spent on helping users with misbehaving jobs. Please note that this does not include any system level maintenance, since that is typically handled by an independent IT team in a uniform way across the local infrastructure.

Since there are only a few Glidein Factory instances, there is instead very little variation between them. For consistency, we will skip configuration and operational procedures for these as well. Here the operational cost tend to be higher, with the bulk of the effort going into adapting to the ever-changing environment of the Grid, and tends to scale linearly with the number of independent Grid sites used for resource provisioning. Again, system level maintenance is not factored in.

4 MOVING TO THE CLOUD

As described in the previous section, the glideinWMS architecture calls for a clear separation of duties between the Glidein Factory and the VO Frontend. When we decided to add support for Cloud resource providers, we were thus required to decide which one would own the system image.

Moreover, one has to decide if the pilot processes will run as a superuser, e.g. UNIX root, or as a non-privileged user. On the Grid, we manage to run securely even without superuser privileges, but we do sacrifice some flexibility in the process.

In this section we provide the three options that we consider feasible, together with advantages and disadvantages of each option.

4.1 Image Owned by the Glidein Factory and Pilot Running as a Non-privileged User

One option is to give full control over the system image to the Glidein Factory, and to drop privileges early in the process, thus running the pilot as a non-privileged user. In this scenario, we basically mimic the Grid operational mode.

This choice would provide several advantages:

- The maintenance and security patching of the system image is maintained by the Glidein Factory operators, likely leading to a much better security posture and lower total operational costs compared to the other alternatives.
- The absence of non-system services running with superuser privileges further minimizes security risks.
- The absence of runtime changes to the system level settings allows for certification of the system images.
- The VO Frontend administrators are completely unaware of the difference between the Grid and the Cloud resources, making life much easier for them.

The disadvantages of this choice are instead:

- Reduced flexibility for the Glidein Factory operators, since any change to the system level requires a new image. This would likely require a separate image for each resource type, increasing the operational costs.
- The inability for the VO Frontend administrators to influence the operating system in any way.
- Inability of the pilot to use all the OS features for user job control and monitoring.

4.2 Image Owned by the Glidein Factory and Pilot Running as Privileged User

Another option is to keep the pilot running as a privileged user, while still leaving the full control over the system image in the Glidein Factory. Please notice that this implies that the VO Frontend can provide configuration scripts that would run as the superuser; this is acceptable, since the resource was provisioned using credentials provided by the VO Frontend.

This choice would provide several advantages:

- Similar to the previous option, the maintenance and security patching of the base system image is maintained by the Glidein Factory operators, with all the same advantages.
- The VO Frontend administrators can change system level settings to suit their needs.
- The Glidein Factory operators retain the ability of changing system level settings at run time.
- The pilot has access to all the OS features, allowing it to behave as a full featured batch system daemon.

The disadvantages of this choice are instead:

- Lowered security posture due to runtime changes to the system level settings.
- The VO Frontend administrators are responsible for the maintenance and security patching of any system level service they add, likely resulting in higher total operational costs.
- The VO Frontend still cannot provide the OS of his choice.
- Possibly increased network related operational costs due to runtime loading of configurations.

4.3 Image Owned by the VO Frontend

The third option is to give full control over the system image to the VO Frontend. However, since the VO Frontend should not be aware of the low level details of the provisioned resource, the Glidein Factory will have to contextualize at run time the system image, which is equivalent to running the pilot as a superuser at least a fraction of the time. We thus assume the pilot will indeed be running as a superuser all the time; dropping privileges at the last minute is of course possible, but does not significantly change the outcome.

This choice would provide several advantages:

- The VO Frontend administrators are now able to provide the OS of their choice.
- The Glidein Factory operators retain the ability of changing system level settings at run time.
- The pilot has access to all the OS features, allowing it to behave as a full featured batch system daemon.

The disadvantages of this choice are instead:

- The maintenance and security patching of the system image is maintained by the VO Frontend operators, requiring a higher level of expertise there, and likely leading to a higher total operational cost.
- Lowered security posture due to runtime changes to the system level settings.
- Possibly increased network related operational costs due to runtime loading of configurations.

4.4 Selecting an Option

As can be seen, no option is a clear winner; there are advantages and disadvantages to every approach. The glideinWMS project is thus considering supporting all of them. However, for our initial release we decided to pick one option over the others and support that option only; later releases will likely support the other as well.

We have chosen to support the second option, i.e. the one with the image owned by the Glidein Factory and pilot owned by a superuser. This option provides still enough flexibility for the VO Frontend administrators while keeping both the security risks and the operational costs reasonably low.

The reasons for this choice are:

- The current VO Frontend administrators are content with the current operational model, where they do not have to worry about low-level details. Delegating image maintenance to the current user base may not be a good fit.
- As the user community has origins on the Grid, the number of OS versions required by our user community is limited, so maintaining a couple system images at the Glidein Factory side can be more efficient than having each VO Frontend administrator maintaining its own.
- Some user communities have expressed the desire of obtaining access to system level configurations, in particular to enable customized remote file system mounts. This would not be possible with option one, i.e. with pilot dropping privileges immediately after boot. Moreover, merging all requests from all the VO Frontend administrators on the Glidein Factory side into

a single system image does not seem neither flexible nor particularly secure.

- The major drawback of option two over option one is the potentially lower security posture, due to the pilot's ability to change system level settings at runtime. We plan to mitigate this by ensuring that the privileges are dropped for all operations that do not explicitly request superuser privileges.

5 FUTURE WORK

The glideinWMS project is in the process of releasing the first Cloud-enabled version. While a lot of work has gone into the hardening of the code to make it a candidate for production use, we are fully aware that it needs to be battle tested by real users in order to show all the strengths and weaknesses of our choices.

We are thus eager to get the new release in the hands of actual users, and collect their feedback. We hope that the choices we made will prove to be the right ones, but we are open to re-evaluating any choice that is shown to be suboptimal if not outright wrong.

6 CONCLUSIONS

Moving from the Grid to the Cloud brings along the problem of the creation and maintenance of the system image used to configure the provisioned resources.

Most Grid user communities do not want to abandon the batch system paradigm of the Grid, so a pilot WMS is often the best way forward, since it preserves the batch system experience for the final users by moving the system image handling into a separate, dedicated service.

The glideinWMS system goes a step further, and further abstracts the resource contextualization from the resource provisioning logic. In the Cloud use case, this allows for the offloading of the system image maintenance to a service not maintained by the VO, into the so called Glidein Factory, thus drastically reducing the VO's operational cost. The major drawback of this approach is the inability of the VO administrators to fully customize the system image, but we believe that the limited customization of the proposed solution still satisfies the needs of our users.

ACKNOWLEDGEMENTS

This work is partially sponsored by the US National Science Foundation under Grants No. OCI-0943725 (STCI), PHY-1104549 (AAA), and PHY-0612805 (CMS Maintenance & Operations), and the US Department of Energy under Grants No. DE-SC0002298 (ANDSL) and DE-FC02-06ER41436 subcontract No. 647F290 (OSG).

REFERENCES

Mell, P. and Grance, T. (2011). The nist definition of cloud computing. Technical Report Special Publication 800-145, NIST.

Sfiligoi, I., Bradley, D., Holzman, B., Mhashilkar, P., Padhi, S., and Wurthwein, F. (2009). The pilot way to grid resources using glideinwms. In *Computer Science and Information Engineering, 2009 WRI World Congress on*, volume 2, pages 428 –432.

Sfiligoi, I., Würthwein, F., Andrews, W., Dost, J. M., MacNeill, I., McCrea, A., Sheripon, E., and Murphy, C. W. (2011a). Operating a production pilot factory serving several scientific domains. *Journal of Physics: Conference Series*, 331(7):072031.

Sfiligoi, I., Würthwein, F., Dost, J. M., MacNeill, I., Holzman, B., and Mhashilkar, P. (2011b). Reducing the human cost of grid computing with glideinwms. In *CLOUD COMPUTING 2011, The Second International Conference on Cloud Computing, GRIDs, and Virtualization*, pages 217 –221.