Cloud Bursting with GlideinWMS: Means to satisfy ever increasing computing needs for Scientific Workflows

Parag Mhashilkar [1, a], Anthony Tiradani [a], Burt Holzman [a], Krista Larson [a], Igor Sfiligoi [b], Mats Rynge [c]

[a] Fermi National Accelerator Laboratory
P O Box 500, Batavia, IL – 60510, US

[b] University of California at San Diego
Department of Physics, 9500 Gilman Drive, La Jolla, CA 92093-0354

[c] Information Sciences Institute (ISI)
4676 Admiralty Way, Suite 1001, Marina del Rey, CA 90292

E-mail: parag@fnal.gov, tiradani@fnal.gov, burt@fnal.gov, klarson1@fnal.gov, isfiligoi@ucsd.edu, rynge@isi.edu

Abstract. Scientific communities have been in the forefront of adopting new technologies and methodologies in the computing. Scientific computing has influenced how science is done today, achieving breakthroughs that were impossible to achieve several decades ago. For the past decade several such communities in the Open Science Grid (OSG) and the European Grid Infrastructure (EGI) have been using GlideinWMS to run complex application workflows to effectively share computational resources over the grid. GlideinWMS is a pilot-based workload management system (WMS) that creates on demand, a dynamically sized overlay HTCondor batch system on grid resources. At present, the computational resources shared over the grid are just adequate to sustain the computing needs. We envision that the complexity of the science driven by "Big Data" will further push the need for computational resources. To fulfill their increasing demands and/or to run specialized workflows, some of the big communities like CMS are investigating the use of cloud computing as Infrastructure-As-A-Service (IAAS) with GlideinWMS as a potential alternative to fill the void. Similarly, communities with no previous access to computing resources can use GlideinWMS to setup up a batch system on the cloud infrastructure. To enable this, the architecture of GlideinWMS has been extended to enable support for interfacing GlideinWMS with different Scientific and commercial cloud providers like HLT, FutureGrid, FermiCloud and Amazon EC2. In this paper, we describe a solution for cloud bursting with GlideinWMS. The paper describes the approach, architectural changes and lessons learned while enabling support for cloud infrastructures in GlideinWMS.

1. Introduction

Distributed and High Throughput computing in form of computing grids has been widely adopted by widespread scientific communities with high computing demands, such as high-energy physics (HEP). While computing grids enable pooling of resources across different administrative domains with

[1] To whom any correspondence should be addressed.

relative ease, they also bring several challenges for the users in the management of these resources. Several communities, or Virtual Organizations (VO), in the Open Science Grid (OSG) [1] and the European Grid Infrastructure (EGI) have effectively used GlideinWMS to circumvent these challenges. Section 2 gives an overview of GlideinWMS, a pilot-based workload management system.

Over the past few years, complexity of the science and the demand for computational resources to process scientific data has been steadily growing. While the resources currently available on the grids are sufficient to sustain the demand for the time being, scientific communities are on the look out for new avenues to increase the computing capacity. As the use of commercial and scientific clouds begin to grow and mature, several communities like CMS, Atlas and Intensity Frontier (IF) experiments are investigating computing clouds as a viable option to fuel their computing needs.

Computing resources provided by the clouds are similar those on the grids. GlideinWMS takes advantages of these similarities to interface with the computing clouds, both, commercial as well as research clouds to run scientific workflows. Section 4 describes the approach used to interface GlideinWMS with Computing Clouds like Amazon Web Services AWS [2], Eucalyptus [3], OpenStack [4] and OpenNebula [5] and. Section 5 describes our experience running on these clouds and the challenges faced so far. We discuss future work in section 6 and we conclude in section 7.

2. GlideinWMS

A pilot-based WMS is a pull-based WMS that creates an overlay pool of compute resources on top of the grid. Several pilot-based WMS infrastructures [6] are used by different VOs in OSG and EGI. GlideinWMS is a pilot-based WMS that creates on demand a dynamically sized overlay HTCondor batch system [7][8] on grid and cloud resources to address the complex needs of VOs in running application workflows. This section describes HTCondor and GlideinWMS.

2.1. HTCondor

HTCondor (formerly known as Condor) started as a batch system for harnessing idle cycles on personal workstations [9]. Since then it has matured to become a major player in the compute resource management area.

An HTCondor pool is composed of several logical entities, as shown in figure 1:

- The central manager acts as a repository of the queues and resources. A process called the *"collector"* acts as an information dashboard.
- A process called the *"startd"* manages the compute resources provided by the execution machines (worker nodes in the diagram). The *startd* gathers the characteristics of compute resources such as CPU, memory, system load, etc. and publishes them to the collector.
- A process called the *"schedd"* maintains a persistent queue for jobs submitted by the users.
- A process called the *"negotiator"* is responsible for matching the resources to user jobs.

The communication flow in HTCondor is fully asynchronous. Each of the *startds* and *schedds* advertise the information to the collector independently. Similarly, the *negotiator* starts the matchmaking cycle using its own timing cycle. The negotiator periodically queries the *schedds* to get the characteristics of the queued jobs and matches them to available resources. All matches are then ordered based on user priority and communicated back to the *schedds* that in turn transfer the matched user jobs to the selected *startds* for execution. To fairly distribute the resources among users, the negotiator tracks resource consumption by users and calculates user priorities accordingly. With these characteristics, HTCondor is an excellent choice for a pilot-based WMS; all a pilot job needs to do is configure and start a *startd* that reports as an idle compute resource to the collector.

2.2. GlideinWMS overview

GlideinWMS [10] is a pilot based workload management system that is built on top of HTCondor. Functionally, it can be organized into following logical architectural entities as shown in figure 2.

- GlideinWMS Provisioning Service consists of Glidein factory and VO Frontend services provides following functionality -

- VO frontend monitors the status of jobs in the HTCondor Scheduler and regulate the number of pilot job requests made to the factory.
- Glidein factory acts on the frontend requests and submits pilot jobs to the grid/cloud sites using HTCondor-G [11] as a grid submission tool.
- HTCondor collector is used as a communication dashboard for message exchange between different GlideinWMS services.

• Glidein Service is a lightweight job that runs on the worker node and bootstraps the environment on the worker node to run actual user jobs.

The GlideinWMS architecture supports multiple glidein factories and frontends working together through a common HTCondor collector. This separation of tasks allows for better scalability and separation of operational responsibilities between grid debugging and policy management [12].

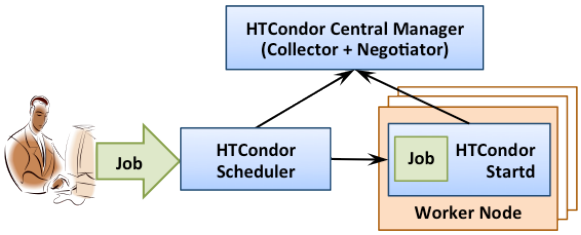

Figure 1. HTCondor architecture overview

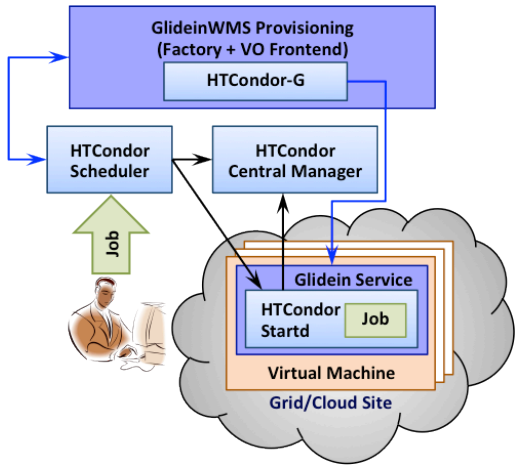

Figure 2. GlideinWMS architecture overview

3. Cloud Computing

Architecturally, cloud computing is a natural evolution of grid computing and in many ways resembles grid computing [13] but with a pay-as-you-use model. Figure 3 shows a simplified version of provisioning of virtual machines in the cloud. In the case of grids, computing resources primarily consist of worker nodes provisioned by the grid middleware to run one or more computational units. With the technological advancements made in recent years, worker node machines today are much cheaper and powerful enough to simultaneously run several complex computational units. Cloud computing uses different virtualization techniques as an economic way of fragmenting powerful machines and to increase their utilization. This makes cloud computing a possible solution for providing flexible, on demand computing infrastructure for a number of applications.

In the case of clouds, several servers work together to dynamically provision, manage and monitor virtual machines (VM). Users typically develop their own virtual images or use pre-existing virtual images provided by the cloud service provider. Multiple instances of a single virtual image can be launched at the same time to provide an independent but consistent environment of a virtual cluster to perform computation.

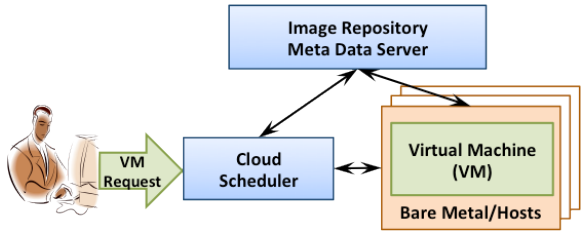

Figure 3. Provisioning VM in Cloud

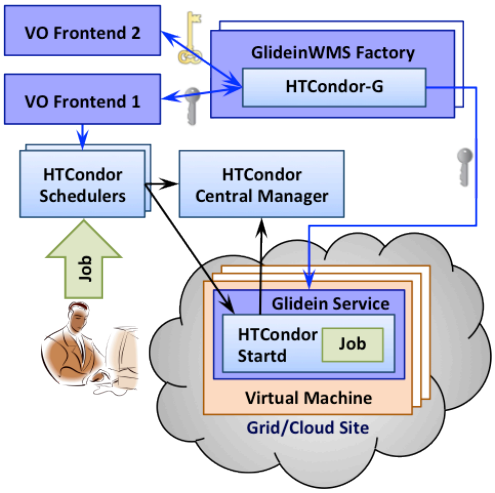

Figure 4. Interfacing GlideinWMS with Cloud

4. Interfacing GlideinWMS with Computing Clouds

Today, GlideinWMS is the leading pilot based WMS used by the scientific communities in the OSG and the EGI. For the past several years, GlideinWMS has been successfully used to provision resources on the OSG, EGI and Nordu-Grid sites. In order to support its broader user base, GlideinWMS has been extended to add support for cloud resources. Figure 4 shows the use case of GlideinWMS interfacing with a grid or an EC2 enabled cloud site. Based on the operational policies either a VO or the factory operator could be responsible for maintenance of the VM image.

Grid sites use x509 credentials for authentication and authorization between users and services. Clouds, however, lack similar federated identity management. Different clouds support different authentication mechanisms, but in most cases use a username-password based authentication mechanism or its variation in some form. In order to support clouds, GlideinWMS was extended to support different security credentials types to work with both grid and cloud sites. In addition to different supported credential types, credentials accepted at one cloud site differ from those at other cloud site. To cleanly support different security credential types, GlideinWMS introduced the concept of *Trust Domain*. A trust domain forms a logical group of sites that support a common set of credential types. A frontend needs to support at least one credential for a trust domain to request glideins at any of the sites of that trust domain. The frontend then uses the trust domain as one of the matching making criteria for requesting glideins.

4.1. Glideins in clouds v/s in grids

The table below highlights these differences to consider when running workflows in grids v/s running them in the clouds using GlideinWMS.

Cloud Site	**Grid Site**
Glidein is a service that runs inside the VM launched by HTCondor as a job using EC2 API. Requires lightweight glidein services installed in the VM image.	Glidein is an HTCondor job that runs inside a worker node. This does not require any GlideinWMS software to be installed on the worker node.
Cloud provider facilitates hosting the VM Image provided by the VO. VO manages the image, i.e. software stack installed & root access to the worker node. Based on the policy, cloud provider can also be the VM maintainer/administrator.	Grid site admin manages the Worker Node, i.e. software stack installed & root access to the worker node.

Table 1. Glideins in Clouds v/s in Grids

As seen in the table above, these differences are either in the communication layer between GlideinWMS services or they affect how the GlideinWMS services are operated. The end user running jobs on grid and/or cloud resources is completely shielded and would not notice any functional differences between the resources provisioned in the cloud versus those in the grids.

5. Production in Clouds: Experience & Challenges

Several scientific communities have only recently started using clouds with GlideinWMS provisioning as an option to increase the computing capacity. Figure 5 shows running VMs (requested by glideins) and the batch slots provisioned (claimed by user jobs) by GlideinWMS for CMS in the (High Level Trigger) HLT cloud. Figure 6 shows similar plots from use of GlideinWMS by NOvA, one of the Intensity Frontier's experiments at Fermilab using FermiCloud resources. In both these cases, GlideinWMS team worked closely with the HTCondor team to address several provisioning and scalability issues. Based on our experience running production in HLT cloud [14] and FermiCloud [15] there are several challenges that need to be addressed before computing in clouds matures to the same scale as in grids today.

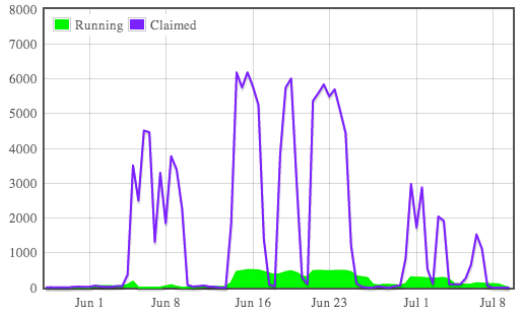

Figure 5. GlideinWMS in HLT

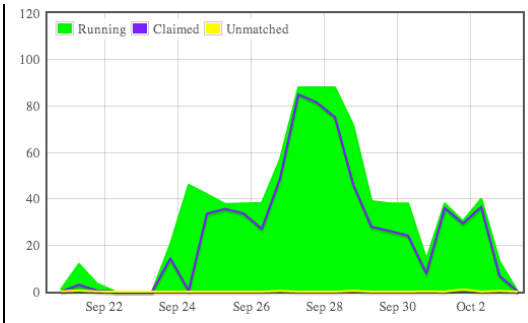

Figure 6. GlideinWMS in FermiCloud

5.1. Challenges

This section describes some of the challenges encountered while provisioning cloud resources using GlideinWMS

5.1.1. Different Implementations of Cloud Middlewares and EC2 APIs

Unlike in grid computing where WMS interfaces to a site via one of the standard grid protocols, such as GRAM, CREAM and ARC, each cloud middleware supports their own proprietary protocol. Supporting different cloud protocols to provision resources in any WMS like GlideinWMS soon becomes un-scalable. Amongst the big commercial clouds like Amazon's AWS, Microsoft's Azure Cloud and Google's Compute Engine, AWS is one of the oldest and the largest. However, most scientific communities are also interested in open source solutions like OpenStack, OpenNebula, Eucalyptus, Nimbus [13], etc. The open source cloud solutions do support a subset of Amazon's EC2-Query API in addition to their proprietary protocol. However, each of these clouds implements different subsets of the EC2-Query API with varying feature sets. EC2-Query API implemented by Amazon is merely a community-accepted convention rather than a well-governed standard. Since GlideinWMS uses HTCondor to communicate with the clouds (via the EC2-Query API) to provision resources, its functionality is limited to the subset of Clouds supported by HTCondor.

5.1.2. Different Implementations of EC2 USER DATA

Cloud middleware provides a means to pass any user specific information to the cloud VM. In the EC2 world this is referred to as *EC2 USER DATA*. GlideinWMS uses *EC2 USER DATA* to pass contextualization data, including security credentials to the glidein service. Core GlideinWMS services are relatively well shielded from differences in the EC2-Query API implementations of

different cloud middleware. But, GlideinWMS is greatly impacted by the different methods employed by the middleware to pass the *EC2 USER DATA*. For example, Amazon AWS, OpenStack and Nimbus have a Meta Data server that the glidein service access to get the *EC2 USER DATA*. However, there are differences in the means of accessing the Meta Data service. On the other hand, OpenNebula does not have a Meta Data server and passes this information to the VM using a mounted disk. The glidein service needs to account for these differences before supporting a new cloud middleware.

5.1.3. Cloud Scheduler Scalability

The GlideinWMS team has most operational experience working with Amazon EC2 and HLT's OpenStack cloud. HTCondor periodically polls (every 5 minutes by default) the cloud scheduler to determine the status of the provisioned VM. Amazon schedulers are known to scale quite well. However, we noticed heavy load and scaling issues when scheduling around 500 VMs in the Nova scheduler of OpenStack. One alternative is to use a different scheduler with OpenStack that scales. We also have limited experience running NOvA experiment workflows in FermiCloud's OpenNebula cloud deployed at Fermilab. Here we did not notice any issues provisioning 50+ VMs using GlideinWMS. The plan in near future is increase this number to around 2000 VMs.

5.1.4. Virtual Machine Resource Clean-up

GlideinWMS provisioned resources have a set lifetime and are periodically relinquished using predefined rules. For example, lifetime of a glidein in grids is limited by a predefined value or if there is no activity for some time. GlideinWMS uses similar policy in case of cloud VMs. The glidein service shuts down the VM after a pre-configured value or if there is no VM utilization for some time. To handle such a use case, Amazon EC2 supports terminate on shutdown flag to relinquish the cloud resources. However, OpenStack ignores this flag while OpenNebula does not support such a flag. As a result, the VM goes into *SHUTDOWN* state in OpenStack and in *UNKNOWN* state in OpenNebula. In either case, VM stays in the cloud queue and the resources are not relinquished back. The GlideinWMS team is actively working with the cloud middleware developers to provide a consistent approach to handle this use case.

6. Future Work

In order to increase the adaptability of cloud for scientific computing, significant effort is required on all fronts to make the system scale to a comparable level as grids. In this section we list some of the important functionality that needs to be implemented or improved in GlideinWMS.

GlideinWMS needs to support better provisioning mechanisms to take into account varying cost for computing across different types of cloud and grid resources. Currently, a time based provisioning mechanism works reasonably well, however, to utilize other beneficial features of the cloud, like spot pricing, requires a more sophisticated logic.

The provisioning middleware currently does not provide the possibility of fetching output/log files from the VM. An intermediate solution in GlideinWMS to transfer files back to the GlideinWMS Pool using FTP like services is possible, but needs to be implemented on case-by-case basis. A better solution would be for the cloud services to provide API to transfer the files back in a manner similar to the *EC2 USER DATA* Meta data service.

7. Conclusions

Several VOs using GlideinWMS are investigating a viability of running scientific workflows on clouds. GlideinWMS has been extended to support computing on cloud resources that provide Amazon's EC2 compatible APIs. These changes in GlideinWMS shield the end user from the differences between cloud and grid resources. Small-scale production on CERN's HLT cloud and Fermilab's FermiCloud infrastructure using GlideinWMS to provision resources has shown promising results. However, the current infrastructure is missing several important features before a full-scale production use of cloud resources is possible.

Acknowledgements
Fermilab is operated by Fermi Research Alliance, LLC under Contract number DE-AC02-07CH11359 with the United States Department of Energy (DOE).

The paper is partial sponsored by following grants –

- DOE and KISTI under a joint Cooperative Research and Development Agreement CRADA-FRA 2013-0001 / KISTI-C13013
- NSF grants PHY-1120138 and OCI-0943725

The authors would like to thank the HTCondor team for their continuous support to expand the functionality and improve the scalability of provisioning cloud resources using HTCondor.

The authors would also like to thank the HLT cloud at CERN and FermiCloud at Fermilab for their help with the scale testing.

References

[1] R. Pordes, et. al., "The Open Science Grid", Journal of Physics: Conference Series 78, IoP Publishing, 2007 (15pp)

[2] Amazon AWS EC2 Documentation (accessed Oct 14, 2013): http://aws.amazon.com/documentation/ec2/

[3] Nurmi, Daniel, Richard Wolski, Chris Grzegorczyk, Graziano Obertelli, Sunil Soman, Lamia Youseff, and Dmitrii Zagorodnov. "The eucalyptus open-source cloud-computing system." In Cluster Computing and the Grid, 2009. CCGRID'09. 9th IEEE/ACM International Symposium on, pp. 124-131. IEEE, 2009.

[4] Corradi, Antonio, Mario Fanelli, and Luca Foschini. "VM consolidation: a real case based on OpenStack Cloud." Future Generation Computer Systems (2012).

[5] Sotomayor, Borja, Rubén Santiago Montero, Ignacio Martın Llorente, and Ian Foster. "Capacity leasing in cloud systems using the opennebula engine." InWorkshop on Cloud Computing and its Applications, vol. 2008. 2008.

[6] I. Sfiligoi, et. al., "The Pilot Way to Grid Resources Using glideinWMS", CSIE '09 Proceedings of the 2009 WRI World Congress on Computer Science and Information Engineering - Vol 02

[7] HTCondor homepage (accessed on Oct 14, 2013): http://www.cs.wisc.edu/htcondor/

[8] M. Litzkow, M. Livny, and M. Mutka, "Condor – A Hunter of Idle Workstations", Proc. of the 8th Int. Conf. of Dist. Comp. Sys., June, 1988, pp 104-111.

[9] Parag Mhashilkar, Zach Miller, Rajkumar Kettimuthu, Gabriele Garzoglio, Burt Holzman, Cathrin Weiss, Xi Duan, Lukasz Lacinski, "End-To-End Solution for Integrated Workload and Data Management using glideinWMS and Globus Online", presented at Computing in High Energy Physics (CHEP 2012), New York, USA, May 2012

[10] GlideinWMS homepage: (accessed on Oct 14, 2013): http://www.uscms.org/SoftwareComputing/Grid/WMS/glideinWMS

[11] J. Frey et. al., "Condor-G: A Computation Management Agent for Multi-Institutional Grids", Journal of Cluster Computing, 2002, vol 5, pp. 237-246.

[12] Sfiligoi, Igor, Frank Würthwein, Jeffrey M. Dost, Ian MacNeill, Burt Holzman, and Parag Mhashilkar. "Reducing the human cost of grid computing with glideinwms." In CLOUD COMPUTING 2011, The Second International Conference on Cloud Computing, GRIDs, and Virtualization, pp. 217-221. 2011.

[13] Keahey, Kate, Renato Figueiredo, Jose Fortes, Tim Freeman, and Mauricio Tsugawa. "Science clouds: Early experiences in cloud computing for scientific applications." Cloud computing and applications 2008 (2008): 825-830.

[14] Liu, Guoming, and Niko Neufeld. "LHCb online in the cloud: off-site computing resources for the LHCb High Level Trigger." In Real Time Conference (RT), 2012 18th IEEE-NPSS, pp. 1-4. IEEE, 2012.

[15] FermiCloud homepage (accessed on Oct 14, 2013): http://fclweb.fnal.gov/