Grappling Cloud Infrastructure Services with a Generic Image Repository

Javier Diaz, Andrew J. Younge, Gregor von Laszewski, Fugang Wang and Geoffrey C. Fox
Pervasive Technology Institute. Indiana University
2719 East Tenth Street, Bloomington, IN 47408, USA
{javidiaz,ajyounge,gvonlasz,fuwang,gcf}@indiana.edu

ABSTRACT

With the advent of Cloud computing, a wide variety of Infrastructure as a Service models have grown to provide users with one of the greatest benefits of Clouds: a customized system environment. These services, while extremely useful, often suffer from their ability to interoperate and communicate across administratively separate domains. Within FutureGrid, an experimental cloud and grid testbed, we look to mitigate these challenges by providing stratosphere-level image repository and management tools. These systems will enable users to more easily generate images and manage them within an independent image repository, eventually enabling users to take full advantage of both private and public clouds with ease.

1. INTRODUCTION

FutureGrid (FG) [1] provides a capability that makes it possible for researchers to tackle complex research challenges in Computer Science related to the use and security of grids and clouds. These include topics ranging from authentication, authorization, scheduling, virtualization, middleware design, interface design and cybersecurity, to the optimization of grid-enabled and cloud-enabled computational schemes for researchers in Astronomy, Chemistry, Biology, Engineering, Atmospheric Science and Epidemiology. The project team will provide a significant new experimental computing grid and cloud test-bed to the research community, together with user support for third-party researchers conducting experiments on FutureGrid.

As part of the test-bed, we will have pre-installed frameworks exposed through endpoints to provide users an easy access to them. This includes Infrastructure as a Service (IaaS) frameworks such as Nimbus [7], Eucalyptus [8], OpenNebula [6], and OpenStack [3] and look to provide Platform as a Service (PaaS) frameworks and additional services and tools like Unicore [4] and Genesis II [2]. However, we will also provide additional functionality to instantiate and deploy such frameworks on demand through a provisioning subsystem that is called RAIN [9]. In this way, users can get a customized version of a particular software to fit with their experiments requirements.

Although, we will allow users to run their experiments on both virtual and bare-metal systems, most of the frameworks and services that will be offered by FG are based on virtualization technologies or make use of them. Hence, the image management become a key component in the context of this project. For this reason, in this paper, we present the catalog and repository of images that will be provided by FG. These tools offer a unique and common interface that can distinguish image types for different purposes like an specific IaaS framework or just general HPC. Moreover, it will include base images set up by the project team as well as images generated by the users, if they wish to share them. The images will be described with information about the various software installed in them (including versions, libraries, etc), available services, etc. This information will be maintained in the catalog and be searchable by the users and/or other FG services. Users looking for an specific image will be able to discover the available images and find their location in the repository using the catalog interface. In addition, users can also register customized images, share them among other users, and choose any of them for the provisioning subsystem [9].

The development of a customized image repository not only provide functional advantages, but it also provides structural advantages aimed to make a more efficient use of the resources. In particular, we have the possibility to choose the most suitable storage mechanism for FG. We can also maintain specific data that could assist performance monitoring and accounting. Finally, we can use mechanisms that may actually not just store the image, but rather describe how we generate an image. In this way, we can save a significant amount of time and space for both users and system administrators alike.

2. IMAGE REPOSITORY SERVICE

The FutureGrid image repository provides a service to query, store, and update images through a unique and common interface. Nevertheless, it also maintains data related with the usage of the image repository to assist performance monitoring and user accounting.

We have identified some essential requirements that we need to consider in our design. These requirement include a simple, intuitive and user friendly environment, a unified, extensible, integrated and secure system design, and built in fault tolerance with proper accounting and information

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
Copyright 20XX ACM X-XXXXX-XX-X/XX/XX ...$10.00.

tools. Considering these requirements, we have designed a flexible and modular architecture for the FG image repository, see Figure 1.

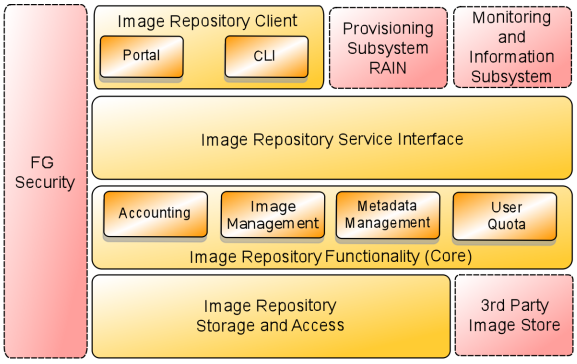

Figure 1: Image Repository Architecture

At the bottom of the Figure 1 we have Storage and Access layer, that defines an interface to create plugins to support different storage systems in a transparent way. Therefore, this layer acts as a bridge between the storage systems and the image repository core functionality. The core is represented by the Functionality layer. This is a centralized component manages the images and the user access. The image management is focused to handle the image files and the associated information (metadata) in such a way that the image catalog is always updated and consistent. This is designed to be independent of the storage system used. Moreover, it registers the image usage and keep track of the images generated using existing ones. In this way, we can know which images are useful and detect security problems identifying those images that need to be updated or even removed. On the other hand, the core also manages users to control the use that they do of the image repository. In particular, it handles image permission, user roles, quotas, etc.

Next to the Functionality layer we have the Service Interface one, see Figure 1. This layer expose all the previously mentioned functionality through a common interface to allow developers to create applications that interact with the image repository. In this sense, we have designed a command line interface (CLI) and a web portal to facilitate the access to the image repository. Other FG subsystems or external systems can get access to the image repository through this interface. As for example, in Figure 1, we show two FutureGrid subsystems, the Provisioning and the Monitoring and Information ones [9].

Finally, the security aspect is an essential component to be considered in the design. Thus, the image repository will provide the security functionality needed to integrate the authentication and authorization with the FG ones (based on LDAP). In this way, we contribute to maintain a single sign on system for FG, avoiding the duplication of services and databases.

3. IMAGE GENERATOR SERVICE

The image generator service is another central component for the FutureGrid imaging system, and is the key feature for adapting workflows with the pretence of a custom user environment to Clouds. The image generator is responsible for taking in user requirements about image size, type, and kind to format a new image that, once vetted and stored, can be deployed on FG hardware. The image generator will start with a base image that is selected by the user. This image is specifically crafted by FG administrators to enhanse stronger security and integration with the rest of FG. It is also designed to be the smallest image footprint possible, to minimize wait time and network traffic when deploying new images. Images are next mounted and the software stack selected is deployed onto the system, along with any other files specified. The image generator then links the new image to BCFG2 [5] and submits it to the image repository.

BCFG2 retains a major component of the image management system. In fact, it does most of the management actions for all the images deployed throughout FG. BCFG2 itself is a critical tool to help system administrators produce a consistent, reproducible, and verifiable description of their environment, and offers visualization and reporting tools to aid in day-to-day administrative tasks. Within FG's BCFG2 deployment, a number of base deployment groups are setup that correspond to the pre-supported OS types added by administrators to the Image Generator. From there, a given image will be assigned another unique group which contains the software stack specified by the user. This allows for all software and files installed on the image to be managed, updated, and verified by BCFG2. This group is created and defined by the image generator before initial deployment.

As described above, there are a number of base images that are supported within FG. These UNIX-based images represent the minimal installation possible within the OS itself. Because many of these image will be leveraged to provide platform-level services, there is no need to add extra packages and bloat to images, especially when the images are to be deployed and migrated throughout FG resources. The base OS is created as a separate .img file by FG administrators with the necessary BCFG2 client pre-installed along with any other monitoring software deemed fit by the FG Performance group.

From a process perspective, a given user selects specific features of the image as needed including the target deployment selection such as OS, and hardware, as well as base software, FG software, Cloud software, user application software and other software. This creates then a *base image*. This base image is than deployed on a test server and updated and checked for security. The result is a deployable image on FG hardware. At time of deployment additional security updates are conducted. It is clear that the time between the creation of a deployable and deployed image has an impact on security. Images found insecure could still be deployed, but the network connectivity to such an image is restricted.

4. CONCLUSION

In this paper we have introduced our design for a generic image repository and image generation tools for use within the FutureGrid testbed. In an attempt to open discussion within the community, we mainly focus on the requirements and design to establish the important features that we look to support. We consider that a key aspect of these image services is the ability to provide a unique and common interface to manage any kind of image, and any kind of cloud infrastructure. Moreover, its design is flexible enough to be easily integrable not only with FutureGrid but also with other

frameworks, can will help alleviate both the user burden of creating and storing images, but also avoid interoperability issues between different Cloud deployments.

5. REFERENCES

[1] Futuregrid portal. http://portal.futuregrid.org, Last access Mar. 2011.

[2] Genesis II webpage. http://www.cs.virginia.edu/∼vcgr/wiki/index.php/The_Genesis_II_Project, Last access Mar. 2011.

[3] Openstack webpage. http://openstack.org/, Last access Mar. 2011.

[4] Unicore webpage. http://www.unicore.eu/, Last access Mar. 2011.

[5] N. Desai, R. Bradshaw, J. Hagedorn, C. Lueninghoener, et al. Directing change using Bcfg2. In *Proceedings of the 20th Large Installation System Administration (LISA) Conference*, pages 215–220, 2006.

[6] J. Fontan, T. Vazquez, L. Gonzalez, R. S. Montero, and I. M. Llorente. OpenNEbula: The Open Source Virtual Machine Manager for Cluster Computing. In *Open Source Grid and Cluster Software Conference*, San Francisco, CA, USA, May 2008.

[7] K. Keahey, I. Foster, T. Freeman, X. Zhang, and D. Galron. Virtual Workspaces in the Grid. *Lecture Notes in Computer Science*, 3648:421–431, 2005.

[8] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov. The Eucalyptus Open-source Cloud-computing System. *Proceedings of Cloud Computing and Its Applications*, 2008.

[9] G. von Laszewski, G. C. Fox, F. Wang, A. J. Younge, A. Kulshrestha, G. G. Pike, W. Smith, J. Voeckler, R. J. Figueiredo, J. Fortes, K. Keahey, and E. Delman. Design of the futuregrid experiment management framework. In *GCE2010 at SC10*, New Orleans, 2010. IEEE, IEEE.