Towards Generic
FutureGrid Image Management
Gregor von Laszewski, Javier Diaz, Fugang Wang, Andrew J. Younge, Archit Kulshrestha, and Geoffrey Fox
Pervasive Technology Institute, Indiana University
2729 E 10th St, Bloomington, IN 47408, U.S.A.
laszewski@gmail.com

ABSTRACT
In this paper, we briefly outline the current design of a generic image management service for FutureGrid. The service is intended to generate, store, and verify images while interfacing with different localized cloud IaaS image. Additionally, we will also use the service to generate images for traditional bare-metal deployments.

Categories and Subject Descriptors
H.3 Online Information Services. H.3.5 [Information Storage and Retrieval] Online Information Services. Data sharing

General Terms
Design, Experimentation.

Keywords

1. INTRODUCTION
FutureGrid (FG) [1] is an experimental test-bed that supports HPC, cloud and grid computing experiments for both application and computer scientists. FutureGrid includes the use of virtualization technologies to allow the support of a wide range of operating systems in order to include a test-bed for various cloud computing infrastructures as a service frameworks. Therefore, efficient management of a variety of virtual machine images becomes a key issue. Current cloud frameworks do not provide a way to manage images for different IaaS frameworks. Instead, they provide their own image repositories, but in general they do not offer services to store meta data to manage different IaaS images in the same repository. This paper presents a generic catalog and image repository to store images of any type and their associated metadata for use within any cloud service.

A number of efforts exist to provide image repositories as part of IaaS/PaaS Cloud frameworks. This includes Nimbus Eucalyptus, OpenStack, OpenNebula, AbiCloud, Amazon Web Services, and Windows Azure. Our work differs from these technologies as we provide an integrated service that overarches the cloud IaaS frameworks installed on FG which enables storing and organizing of images from multiple cloud efforts in the same repository. The images will be described with information about the software stack installed on them. This will include versions, format, libraries, and available services. This information will be maintained in the repository catalog and can be searched by users and authorized FutureGrid services. Instead of just storing images, we will also provide the ability of generating images on-demand based on generic image generation descriptions. This is going to be one of the important features of the repository, as it will enable us to reduce disk usage significantly. Moreover, it will improve our security requirements as this mechanism can also be used to allow images to be updated through scheduled or triggered events.

Images will be augmented with metadata, thereby enabling users to identify the software accessible on the images with ease. Upon a query by users, images will be generated if they are not yet available in the repository. Usage of the images is recorded and images that are rarely used may be purged from the system.

In our design the access to the images can be restricted to single users, groups of users or system administrators. In many current cloud frameworks (including FG), access to images is either local to a user or global to the entire community. The design of the repository will provide access through convenient API's, a command line, an interactive shell, and a REST service. Other cloud frameworks could integrate with this image repository by accessing it through a standard API that we will provide.

2. Design
The FG image management processes are supported by a number of tightly-coupled services essential within FG. We distinguish several major services (see Figure 1): image repository, image generator, image deployment, image verifier, and an experiment management framework. Together these services will implement the following functionality:

a) Image Generation: We support the generation of images that are independent from a given deployment. In addition, we are going to integrate a verification process that improves the trust in this image. This can be achieved by two processes: (1) the user works directly with FG staff to generate an image, as this process may be time consuming and (2) the automatic generation of images based on templates, previously verified images, and a closed controlled workflow that allows the generation of images with well defined restrictions and additions. Restrictions include the access to root in case the image is deployed in bare-metal fashion, while additions include the integration of, for example an ssh daemon that allows authorized user access to be specified by the image creator.

b) Image Verification: In future we will integrate an automatic verification of images. Only if the image passes predefined tests it is marked as deployable. We will control parameters that issue this test in a periodic fashion. The time period is adjustable and...
can be set by the image creator or the FG operations team. If a user requests an image that is not yet verified, the user will be notified, and the image deployment will not continue until the image has not been approved. Hence, verification takes place several times on an image: at time of generation, before and after the deployment, and once a time threshold is reached.

c) Image Sharing: In order to be able to share images we are providing a repository that will be accessible by the users of FG. Access to images can be restricted access by groups, users and project. Quota management will be used in order to restrict the size of the overall repository and enforce equal sharing of resources.

d) Image Deployment and Provisioning: As a critical component of this work, images need to be deployed on various IaaS frameworks supported within FG. If possible, we will include on-demand transformation workflows to derive images running on different IaaS frameworks. Thus the deployment of an image may activate automatically its creation from a common base. This also addresses the issue of limited storage space and quota not only within the repository, but also in the systems where we deploy the images. We will integrate a time-to-live function that is coupled with our distributed image cache service and actively reduce the storage of outdated, obsolete, and rarely used images. In case an image is frequently used we will attempt not to recreate it with the workflow, but cache this image in the distributed storage to guarantee fast access.

e) Accounting: In order to guide our system with information about usage and utilization we will integrate an accounting mechanism that reports which users use what images and when. This information can be used to meaningfully decide if an image can be deleted, needs to be kept, or must be generated in advance.

f) Authentication and Authorization: Just as in any other HPC environment we need also to think about authentication and authorization in images provisioned in FG. It naturally needs to be integrated with our general policies of account management and project approval. We must gracefully deal with a change in user’s privileges at any arbitrary point in time by integrating certificate revocations and validation of valid accounts. In addition, we must be aware of project based restrictions and allow user to create selective policies for authorization based on project participation. Therefore, we will consider the following four user groups: (a) a single users who create images for themselves, (b) a group of users who share the image amongst themselves, and (c) system administrators who maintain the images for the standard FG resource deployments such as HPC.

h) Reproducibility: An important goal of FG is the creation of an environment supporting a reproducible experiment management service [2]. This means that a user should be able to describe an experiment in such a fashion that it can be recreated by himself or by someone else. The image management service we develop is an important ingredient in this goal as it provides us with the elementary capabilities for image support as part of sophisticated image management.

3. Implementation and Development

We are gradually implementing our design goals and have provided an implementation of an initial image repository. The repository is developed in python and provides at this time command line interfaces and APIs to interface with it. A REST service is under development. At this time our development focus is to integrate the image generation, the repository, and the “raining” of images on FG services and resources. Raining refers to the process of provisioning the appropriate images on either IaaS or HPC environments.

4. Impact on XD

Due to the general nature of the FG image repository it could be made available also to users of XD. Moreover, we will be integrating and leveraging from the XD TAS project. This will especially include the investigation of specific auditing metrics to report usage to the various user communities. However this work will go beyond the representation and focus on the integration of novel trigger services into FG to facilitate better storage and network usage. Hence, we see a valuable synergistic activity arising between other XD projects and FutureGrid.

5. Status

A prototype is available. The development of the image repository is performed as part of the FutureGrid project and the code snapshots are available in the FG code repository.

6. REFERENCES


Figure 1: Image Management Processes