Towards Cloud Deployments using FutureGrid

Javier Diaz, Gregor von Laszewski, Fugang Wang, Geoffrey C. Fox
Indiana University, Bloomington, IN 47408
laszewski@gmail.com

***Abstract*— In this document we briefly outline some differences between IaaS frameworks Eucalyptus, OpenNebula, OpenStack and Nimbus. We provide also an overview how platforms such as Amazon, Azure, and Google provide additional services to provide more convenient platforms for its users. We than present an overview of what FutureGrid currently provides while also focusing on opportunities to leverage from our image generation and management tool to utilize several of the IaaS frameworks.**

***Index Terms*—cloud, grid, Nimbus, Eucalyptus, OpenStack, OpenNebula, RAIN, FutureGrid**

I. Introduction

Cloud computing has become an important driver for delivering infrastructure as a service to users that demand the creation of environments that are customized to their service needs. This not only includes the instantiation of a service, but often the creation of a suitable software stack in which such services are deployed and delivered. Furthermore, we observe the trend that a platform is delivered as a service to its customers that hides the at times very complex task of creating infrastructures suitable for us. Together IaaS and PaaS can provide potent solutions not only to business users, but also to the educational and scientific computing communities. While on the one hand one can imagine cloud and grids to support the most challenging scientific research problems posed by a small number of dedicated scientist, such environments will also be able to support what is today termed the "long tail of science", that is many thousands of scientific users with modest or moderate computing needs that do not necessarily require the presence of a petaflop capable agglomerations of dedicated compute resources.

To evaluate how we need to move forward, we have to start first to analyze some existing IaaS frameworks and identify useful IaaS and PaaS solutions for deployment in a cloud for scientists.

The paper is structured as follows. First we will present a an overview what and why FutureGrid has offered a set of current services to its community. We will than go in much more detail to identify differences between different IaaS offerings.

Then we will provide our thoughts on providing PaaS offerings attractive for our user communities. Next we discuss what implications FG currently provides and also focus on describing our image management tools that allow to be somewhat agnostic towards IaaS frameworks and provide pathways to create images for each of them based on a common base image description.

The later is of special interest as at this time many of the Cloud frameworks are still under heavy development and pathways to utilize multiple of them are of current interest.

II. FutureGrid

In order to test out some of the cloud offerings and to identify what kind of applications benefit from clouds FutureGrid offers possibilities to explore them. The FutureGrid project is sponsored by NSF and includes partners from Indiana University, University of Chicago, University of Florida, San Diego Supercomputing Center, Texas Advanced Computing Center, University of Virginia, University of Tennessee,University of Southern California, Dresden, Purdue University, and Grid 5000.

It has a set of distributed resources among its sites totaling about 5000 compute cores. Resources include a variety of different platforms allowing for interesting heterogeneous distributed compute, network, and storage resources while at the same time allowing to unify resources and services for interoperability and scalability experiments as requested by its user communities. As such FG provides a fertile base environment to explore a variety of IaaS and PaaS offerings.

Currently FutureGrid provides a variety of variety of services. When deciding which services to offer we have based our decision on information that we gathered through our web portal as part of an integrated project proposal and approval process. Each project had the choice to vote and list technologies that were relevant for the execution of their projects. The result of this information is depicted in Figure 1

The FutureGrid project is funded by the National Science Foundation (NSF) and is led by Indiana University with University of Chicago, University of Florida, San Diego Supercomputing Center, Texas Advanced Computing Center, University of Virginia, University of Tennessee, University of Southern California, Dresden, Purdue University, and Grid 5000 as partner sites. This material is based upon work supported in part by the National Science Foundation under Grant No. 0910812.

and Figure 2. We observed the following:

1. Nimbus and Eucalyptus were requested the most. This may not be that surprising as we made most advertisement for these systems and initially recommended them for educational class projects on FG.
2. High Performance Computing was requested as third highest category. This is possibly motivated by our affiliation with traditional HPC and Grid communities as well as the strong ties to XSEDE.
3. Hadoop and map reduce was requested for about 36.78% of all users. This number is higher than the once reported in the figure, as we combined the values for Hadoop and MapReduce while only considering unique entries.
4. We saw recently a surge in requests for OpenStack and OpenNebula as both environments are quite popular due to different reasons. OpenStack has just become one of the preferred open source solutions for cloud computing within a large number of companies, but also within the research community. OpenNebula is quite popular as part of the European Cloud efforts but has gained substantial backing also by US projects such as Clemson University and Fermi Laboratory as part of their cloud strategies.
5. Not surprisingly the largest contingent of our users are technology experts. In fact, when analyzing the data from our projects many consider themselves as technology investigation although they may have motivating applications from scientific domains. Hence we have corrected our data based on a review done by us as best as we could identify.

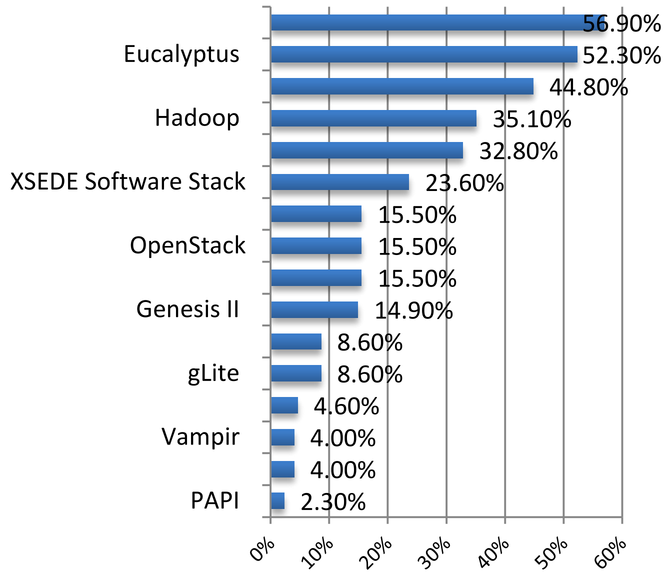


Figure 1: Technology choices made as part of the project application process in FutureGrid. Note that multiple entries could be selected so the total will be more than 100%.

Please note that the data here is only been collected by the project.[1]

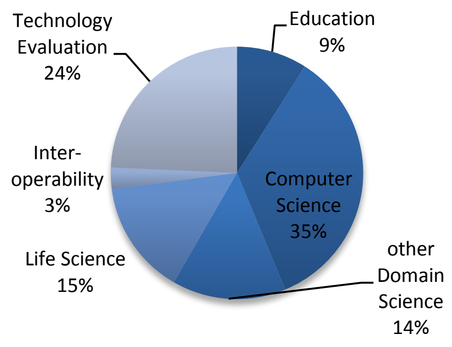
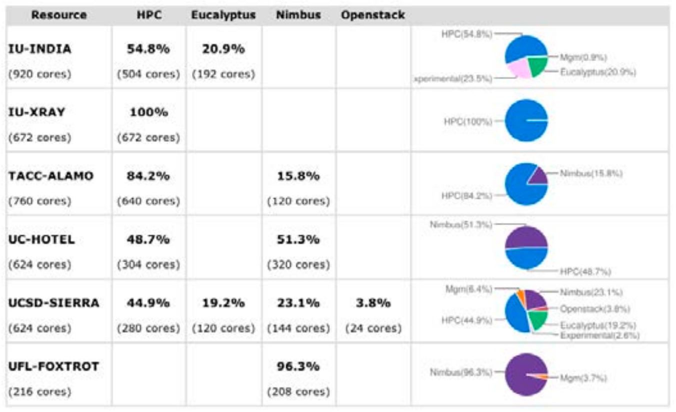


Figure 2: The distribution of the scientific areas that we identified while reviewing the project requests. (Please note that 20 project have yet to be integrated into this data).



Resource	HPC	Eucalyptus	Nimbus	Openstack	
IU-INDIA (920 cores)	54.8% (504 cores)	20.9% (192 cores)			HPC(54.8%) Mgm(0.9%) Eucalyptus(20.9%) xperimental(23.5%)
IU-XRAY (672 cores)	100% (672 cores)				HPC(100%)
TACC-ALAMO (760 cores)	84.2% (640 cores)		15.8% (120 cores)		Nimbus(15.8%) HPC(84.2%)
UC-HOTEL (624 cores)	48.7% (304 cores)		51.3% (320 cores)		Nimbus(51.3%) HPC(48.7%)
UCSD-SIERRA (624 cores)	44.9% (280 cores)	19.2% (120 cores)	23.1% (144 cores)	3.8% (24 cores)	Mgm(6.4%) Nimbus(23.1%) Openstack(3.8%) HPC(44.9%) Eucalyptus(19.2%) Experimental(2.6%)
UFL-FOXTROT (216 cores)			96.3% (208 cores)		Nimbus(96.3%) Mgm(3.7%)

Figure 3: Typical partitioning of FG compute resources by IaaS framework

Based on this analysis we spent our effort to enable such services within FutureGrid. As a result we are currently providing the following partitioning between services as listed in Figure 3. However we have to point out that the number of nodes associated between these services can be changed by request. The reason that OpenNebula does not appear on this chart is that we have not made it officially accessible to our users due to manpower restrictions. However we have conducted scalability experiments (see Section X) that could motivate a possible shift in our current deployment strategy. At present we are working towards making the choice of which IaaS framework to run on the systems more dynamic. For example today I could decide to run Nimbus on the nodes, while tomorrow I could run OpenStack or Open Nebula on them. This helps in evaluating technologies and identifies environments that are best suited for a particular use case.

[1] There is also a slight bias towards the first three technologies as they were part of a required field. In future we will provide a better ay of collecting the information as part of an independent survey.

III. Overview of Cloud IaaS Frameworks

One fundamental concept in cloud computing is based on providing Infrastructure as a Service (IaaS) to deliver resources to customers and users instead of purchasing and maintaining compute, storage, and network. Typically this is achieved through virtual machine offerings. In order to establish such a service, a number of toolkits are available including Eucalyptus, Nimbus, OpenNebula, OpenStack. We will provide a short discussion about these toolkits next and outline some major differences between them.

A. Nimbus

The Nimbus project [1] is working on two products that they term *"Nimbus Infrastructure"* and *"Nimbus Platform"*.

Nimbus Infrastructure: The Nimbus project defines the *Nimbus Infrastructure* to be "an open source EC2/S3-compatible Infrastructure-as-a-Service implementation specifically targeting features of interest to the scientific community such as support for proxy credentials, batch schedulers, best-effort allocations and others." To support this mission, Nimbus is providing their own implementation of
a) a storage cloud that according to the Nimbus project is S3 compatible but is enhanced by quota management b) EC2 compatible cloud services c) a convenient cloud client that is using internally WSRF.

Nimbus Platform: The Nimbus platform is targeting to provide additional tool to its users to simplify the utilization of the infrastructure services and allows integration with other existing clouds including OpenStack and Amazon. To achieve this the following tools have been developed so far: a) cloudinit.d coordinates launching, controlling, and monitoring cloud applications, b) a context broker service that coordinates large virtual cluster launches automatically and repeatably [1, 2]

B. OpenNebula

OpenNebula [3, 4] is an open-source toolkit which allows to transform existing infrastructure into an Infrastructure as a Service (IaaS) cloud with cloud-like interfaces. It has been designed to be flexible and modular to allow its integration with different storage and network infrastructure configurations, and hypervisor technologies [5].

OpenNebula can be used to adapt to organizations with changing resource needs, including addition or failure of physical resources [6]. Some essential features to support changing environments includes live migration and snapshots of VMs [3]. Furthermore, when the local resources are insufficient, OpenNebula can support a hybrid cloud model by using cloud drivers to interface with external clouds. This lets organizations supplement the local infrastructure with computing capacity from public clouds to meet peak demands, or implement high availability strategies.

OpenNebula supports different access interfaces that can be used simultaneously including REST-based interfaces, OGF OCCI service interfaces, and the emerging cloud API standard, as well as the AWS EC2 API service, the de facto cloud API standard.

It also supports cloud federation for scalability, isolation or multiple-site support. Thus, a single access point and centralized management system can be used to control multiple instances of OpenNebula.

The authorization is based on passwords, ssh rsa keypairs, X509 certificates or LDAP. This framework also supports fine-grained ACLs that allow multiple-role support. The authentication.

Finally, the storage subsystem supports any backend configuration, from non shared file systems with image transferring via SSH to shared file systems (NFS, GlusterFS, Lustre…) or LVM with CoW (copy-on-write), and any storage server, from using commodity hardware to enterprise-grade solutions.

C. OpenStack

OpenStack [7] is a collection of open source technologies to deliver public and private clouds. These technologies are OpenStack Compute (called *Nova*), OpenStack Object Storage (called *Swift*), and the recently presented OpenStack Imaging Service (called *Glance*). OpenStack is a new effort and has received considerable momentum due to its openness and may companies supporting this OpenSource effort.

Nova is designed to provision and manage large networks of virtual machines, creating a redundant and scalable cloud computing platform. Swift is used to create redundant, scalable object storage using clusters of standardized servers to store petabytes of accessible data. It is not a file system or real-time data storage system, but rather a long-term storage system for a more permanent type of static data that can be retrieved, leveraged, and then updated if necessary. Glance provides discovery, registration, and delivery services for virtual disk images.

D. Eucalyptus

Eucalyptus [8] promises the creation of on-premise private clouds, with no requirements for retooling the organization's existing IT infrastructure or need to introduce specialized hardware. Eucalyptus implements an IaaS (Infrastructure as a Service) private cloud that is accessible via an API compatible with Amazon EC2 and Amazon S3.

It has five high-level components: Cloud Controller (CLC) that manages the virtualized resources; Cluster Controller (CC) controls the execution of VMs; Walrus is the storage system, Storage Controller (SC) provides block-level network storage including support for Amazon Elastic Block Storage (EBS) semantics; and Node Controller (NC) is installed in each compute node to control VM activities, including the execution, inspection, and termination of VM instances.

IV. Feature Comparison of the IaaS Frameworks

All these IaaS frameworks have been designed to allow users to create and manage their own virtual infrastructures.

Table 1: Feature comparison of IaaS frameworks

	OpenStack	Eucalyptus 2.0	Nimbus	OpenNebula
Interfaces	EC2 and S3, Rest Interface. Working on OCCI □□	EC2 and S3, Rest Interface. Working on OCCI □□	EC2 and S3, Rest Interface □	Native XML/RPC, EC2 and S3, OCCI, Rest Interface □□□
Hypervisor	KVM, XEN, VMware Vsphere, LXC, UML and Microsoft's HyperV □□□	KVM and XEN. VMWare in the enterprise edition. □□	KVM and XEN □	KVM, XEN and VMWare □□
Networking	- Two modes: (a) Flat networking (b) VLAN networking -Creates Bridges automatically -Uses IP forwarding for public IP -VMs only have private IPs □□□	- Four modes: (a) managed (b) managed-novLAN (c) system (d) static -In (a) & (b) bridges are created automatically -Uses IP forwarding for public IP -VMs only have private IPs □□□	- IP assigned using a DHCP server that can be configured in two ways. - Bridges must exists in the compute nodes □□	- Networks can be defined to support Ebtable, Open vSwitch and 802.1Q tagging -Bridges must exists in the compute nodes -IP are setup inside VM □□□
Software deployment	- Software is composed by different component that can be distributed in different machines. - Compute nodes need to install OpenStack software □	- Software is composed by different component that can be distributed in different machines. - Compute nodes need to install OpenStack software □	Software is installed in frontend and compute nodes □□	Software is installed in frontend □□□
DevOps deployment	Chef, Crowbar Puppet □□□	Chef* Puppet* *according to vendor □	no	Chef Puppet □□
Storage (Image Transference)	- Swift (http/s) - Unix filesystem (ssh) □	Walrus (http/s) □	Cumulus (http/https) □	Unix Filesystem (ssh, shared filesystem or LVM with CoW) □
Authentication	X509 credentials, LDAP □□□	X509 credentials □	X509 credentials, Grids □□	X509 credential, ssh rsa keypair, password, LDAP □□□
Avg. Release Frequency	<4month	>4 month □	<4 month	>6 month □
License	OpenSource with Apache license □	BSD license and Commercial, difference between commercial version	OpenSource with Apache license version 2 □	OpenSource with Apache license version 2 □

□□□□□□□□□□□s a positive evaluation. The more checkmarks the better from our point of view.

However, these frameworks have differences that need to be considered when choosing a framework. Thus, we are going to provide a comparison of a selected number of essential features supported by each one. We summarized our findings in Table 1.

Software deployment. This is the first obstacle that we find when we want do deploy our own infrastructure. In our experience the easiest to deploy is OpenNebula because we only have to install a single service in the frontend for a basic configuration while no OpenNebula software is installed in the compute nodes. Nimbus is also relatively easy to install, as only two services have to be configured in the frontend plus the software installation in each compute node. On the other hand, Eucalyptus and OpenStack deployments are quite difficult to achieve due to the entire different components that we need to configure and the different configuration possibilities that they provide.

In addition to a single install we also have to consider update and new release frequencies (see Figure 4). From the release notes and announcements of the framework we observe that major updates happen on a 4 or 6 month schedule, with many release candidates that fix also intermediate issues. Furthermore we observed that the installation deployment depends on scalability requirements and that it is important to note that a deployment lets say for a 4 node OpenStack environment may look quite different from a 60 node installation. Hence, it is important that toolkits providing IaaS can be deployed through configuration management toolkits in order to minimize the effort of repetitive tasks to deploy them on the resources once new versions com out or if a different configuration is set up. Tools such as chef and puppet provide a considerable value add in this regards. Furthermore, they serve as a repeatable "template" to install the services in case version dependent performance comparisons or feature comparisons are conducted by the users.

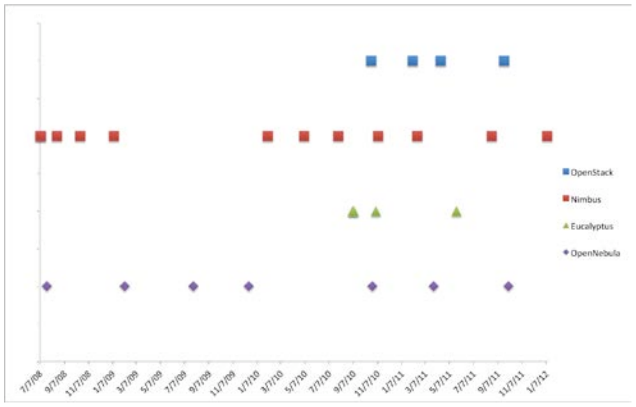


Figure 4: Release frequency of popolar IaaS frameworks

Interfaces. Since Amazon EC2 is a standard de-facto, all of them support the basic functionality of this interface, namely: image upload and registration, and the VM run, describe and terminate operations. However, the OpenStack project noticed disadvantages due to features that EC2 is not exposing. Thus OpenStack is considering to provide interfaces that diverge from the original EC2 standard.

Storage. Storage is very important in cloud because we have to manage many images and they must be available for users anytime. Therefore, most of the IaaS frameworks decided to provide a cloud storage system that can be used not only for internal purposed, but also as an independent product. In the case of Nimbus, it is called Cumulus and it is based on the posix filesystem. Cumulus also provides a plugin that could be used against a variety of storage systems including PVFS, GFS, and HDFS (under a FUSE module). The communication with Cumulus is via http/s. In case of OpenStack and Eucalyptus, they provide a more sophisticated storage system called Swift and Walrus, respectively. Both of them are designed to provide good fault tolerant and scalability. In the case of OpenStack, the images can be stored in the posix filesystem or in Swift. In the first case, images are transferred using ssh while in the second one are transferred using http/s. Finally, OpenNebula does not provide a cloud storage product, but its internal storage system can be configure in different ways. Thus, we can have a shared filesystem between frontend and compute nodes; we can transfer the images using ssh; or we can use LVM with CoW to copy the images to the compute nodes.

Networking. The network is managed differently for each IaaS framework while providing various options in each of them.:

- Eucalyptus offers four different networking modes: *managed*, *managed-noLAN*, *system*, and *static* [9]. In the two first modes, Eucalyptus manages the network of the VMs. They differ in the network isolation provided by vLAN. In the *system* mode, Eucalyptus assumes that IPs are obtained by an external DHCP server. In the static mode, Eucalyptus manages VM IP address assignment by maintaining its own DHCP server with one static entry per VM.
- Nimbus assigns IPs using a DHCP server that can be configured in two ways: centralized and local. In the first case, a DHCP service is used that one configures with Nimbus-specific MAC to IP mappings. In the second case, a DHCP server is installed on every compute node and automatically configured with the appropriate addresses just before a VM boots.
- OpenStack support two modes of managing networks for virtual machines: flat networking and vLAN networking. vLAN based networking requires that you have a vLAN capable managed switch that you can use to setup vLANs for your systems. Flat Networking uses Linux ethernet bridging to connect multiple compute hosts together.
- The OpenNebula network contains the following options: host-managed vLANs where the network access is restricted through vLAN tagging, which also requires support from the hardware switches; Ebtables to restrict the network access through Ebtables rules; and Open vSwitch to restrict network access with Open vSwitch Virtual Switch.

Hypervisors. All of the IaaS frameworks do support KVM and XEN and cover therefore the most popular hypervisors. VMWare is also supported OpenNebula and Openstack. Eucalyptus supports VMWare only in its commercial version. Nimbus does not support VMWare. Additionally, OpenStack also supports LXC, UML and Microsoft's HyperV. This makes OpenStack a quite attractive choice for experimenting with different hypervisors environments.

Authentication. All of the IaaS frameworks support X509 credentials as authentication method for users. OpenStack and OpenNebula also support authentication via LDAP, although is quite basic. OpenNebula also support ssh rsa keypair and password authentication. Nimbus can also provide compatibility with existing Grid infrastructure authentication.

V. Overview of Selected PaaS Frameworks

A. *Platform as a Service*

Under Platform as a Service (PaaS) we elevate services offered to the users beyond the infrastructure and focus on the delivery of a "platform" against which developers can create services while using the features provided by the particular platform. Hence, developers can build applications without installing any tools on their computer and deploy those applications without worrying about system administration tasks.

B. *Azure*

The Windows Azure platform [10] is an Internet-scale computing and services platform hosted in Microsoft data centers. The Windows Azure platform includes the foundation layer of Windows Azure as well as a set of developer services which can be used individually or together. These are Windows Azure (platform for running Windows applications and storing their data in the cloud), SQL Azure (a cloud-based, scale-out version of SQL Server) and Windows Azure AppFabric (a collection of services supporting applications both in the cloud and on premise).

The Windows Azure programming model imposes three rules on applications:

- A Windows Azure application is built from one or more roles. A role defines application files and their configuration. One can define one or more roles for your application, each with its own set of application files and configuration. For each role one can specify the number of VMs, or role instances used as part of the instantiation and execution.
- A Windows Azure application runs multiple instances of each role to provide scalability and fault tolerance.
- A Windows Azure application behaves correctly when any role instance fails because several copies are running. Additionally if the application terminates due to uncontrolled exceptions, Windows Azure will detect this and it will automatically restart the application.

C. *Amazon Web Services*

Amazon [11] is a cloud computing platform composed by a number of products and services. It supports Java, PHP, Ruby and Python languages as well as the .NET platform for application development. Next we mention the most popular services offered by Amazon (see also Table 2).

Services offered by Amazon include computational services such as Elastic Compute Cloud (EC2), which delivers scalable, pay-as-you-go compute capacity in the cloud or Amazon Elastic MapReduce to easily and cost-effectively process vast amounts of data.

Amazon storage services like Simple Storage Service (S3) provide a fully redundant data storage infrastructure for storing and retrieving any amount of data, at any time, from anywhere on the Web.

Database services include both relational and non-relational databases. This services work in conjunction with Amazon S3 and EC2.

Messaging services like the Amazon Simple Queue Service (SQS) make it easy to build workflows between web services.

Other services offered include payment and billing, deployment and management or web traffic.

One advantage of using AWS is its evolving offering of add-ons provided by third –parties. Most notably in the HPC area many are contributors such as Mathworks, Univa/Gridengine, Adaptive Computing, Intel, StackIQ. Cycle Computing. Additionally, open source projects with interesting tools to create clusters such as StarCluster to create clusters using spot pricing used in Bioinformatics and CloudFlu for CFD applications.

Table 2: Amazon Web Service Products

Compute	Elastic Compute Cloud (EC2) (including GPU and extra large instances), Elastic MapReduce, Auto Scaling, Elastic Load Balancing
Content Delivery	CloudFront
Database	SimpleDB, Relational Database Service (RDS), ElastiCache
Deployment & Management	Identity and Access Management (IAM), CloudWatch, Elastic Beanstalk , CloudFormation
Messaging	Simple Queue Service (SQS), Simple Notification Service (SNS), Simple Email Service (SES)
Networking	Route 53, Virtual Private Cloud (VPC), AWS Direct Connect
Payments & Billing	Flexible Payments Service (FPS) , DevPay
Storage	Simple Storage Service (S3), Elastic Block Store (EBS), AWS Import/Export
Support	Premium Support
Web Traffic	Alexa Web Information Service, Alexa Top Sites
Workforce	Mechanical Turk

In the HPC area many contributors including companies such as Mathworks, Univa, Adaptive Computing, Intel, StackIQ. Cycle Computing, put also open source projects with interesting tools to create clusters such as StarCluster and CloudFlu for CFD applications.

D. *Google AppEngine*

Google AppEngine [12] provides a platform to build and host web applications. App Engine includes java and python runtime environments to develop applications. It also provides a Go runtime environment that runs natively compiled Go code. In regards to datastore, it provides a distributed data storage that features a query engine and transactions. This is a non-relational database that can be accessed with GQL (Google Query Language). GQL has SQL like syntax.

Applications run in a secure environment that provides limited access to the underlying operating system. These limitations allow App Engine to distribute web requests for the application across multiple servers, and start and stop servers to meet traffic demands. The sandbox isolates your application in its own secure, reliable environment that is independent of the hardware, operating system and physical location of the web server.

App Engine costs nothing to get started. All applications can use up to 1 GB of storage and enough CPU and bandwidth to support an efficient app serving around 5 million page views a month, absolutely free. When you enable billing for your application, your free limits are raised, and you only pay for resources you use above the free levels.

VI. Supported IaaS and PaaS Frameworks in FutureGrid

As already outlined in Section II FutureGrid provides at this time officially IaaS offerings based on Nimbus, OpenStack and Eucalyptus on various resources. However, We also experimented internally with an OpenStack installation with great success. At this time Nimbus is our preferred IaaS framework due to its easy install, the stability, and the funded support that is provided while including the authors of the Nimbus project as funded partners. This has a positive impact in support tasks, but also in the development of features motivated by FutureGrid. As part of our PaaS offerings we provide various ways of running Hadoop on FG. This is achieved by either using Hadoop as part of the virtualized environment, or exposing it through the queuing system through myHadoop. Twister is contributed through community efforts. Additionally, services such as Unicore and Genesis II are available as part of the HPC services.
At this time we are not supporting any other PaaS offerings such as messaging queues or hosted databases.

Due to the variety of services and limited resources provided in FG it is necessary to enable a mechanism to provision needed services onto resources. This includes also the assignment of resources to different IaaS or PaaS frameworks. We have developed as first step to address this challenge a sophisticated image management toolkit that allows us to not only provision virtual machines, but also provision directly onto bare-metal. Hence we use the term *raining* to indicate that we can place arbitrary software stack onto a resource. The toolkit to do so is called *rain*.

Hence, *rain* makes it possible to compare the benefits of IaaS, PaaS performance issues, as well as evaluating which applications can benefit from such environments and how they must be efficiently be configured. One of the major components that rain includes is our image management service that we explain in the next section in more detail.

VII. Image Management in FutureGrid

The FG image management defines the full life cycle of the images in FG. It involves the process of creating, customizing, storing, sharing and deploying images for different FG environments. Figure 5 shows the high-level image management architecture with its different components and the interfaces that expose the functionality as part of an API, portal and shell command.

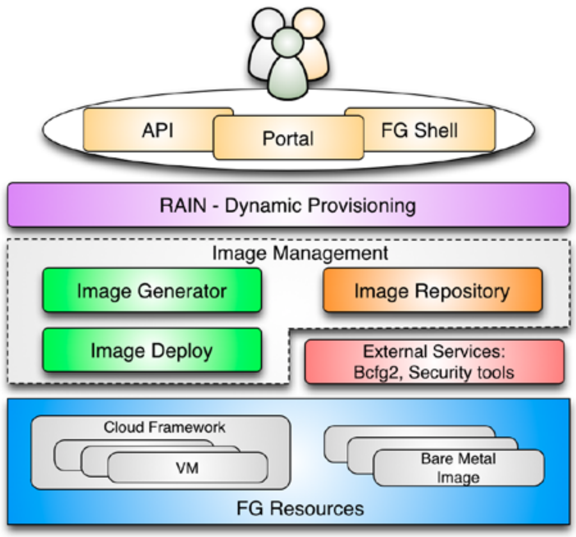


Figure 5: FutureGrid Image Management Architecture.

FG image management core services include three main services:

Image Repository. It provides a service to query, store, and update images through a unique and common interface. Images can be described with information about the software stack, operating system, architecture, etc. This information is maintained in a catalog and can be searched by users and/or other FutureGrid services. Users looking for a specific image can discover available images fitting their needs 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. Security and accounting mechanisms manage the access to the images, the access to administration commands, disk space utilization, and repository usage. This is achieved by managing user roles, quotas, user status and access lists.

Image Generator. It creates images, according to user requirements, that can be deployed in FutureGrid (FG). Since FG is a testbed that support different type of infrastructures like HPC or IaaS frameworks, the images created by this tool are not aimed to any specific environment. Thus, it is at the deployment time when the images are customized to be successfully integrated into the desired infrastructure. This clear separation between image generation and deployment provides a powerful model that allows us to independently increase the OS and infrastructures supported, respectively. Moreover, it reduces the amount of images that we need to manage in the image repository and therefore the disk usage.

Image Deploy. This tool is responsible for customizing images for specific infrastructures and deploying them in such infrastructures. We can distinguish between two main infrastructures types: HPC and cloud. In both cases we need to make some configuration like network IP, DNS, file system table and kernel modules. Additional configurations are performed depending of the infrastructure type. Thus, a deployment into the HPC infrastructure means that we are going to create network bootable images that can run in bare metal machines. On the other hand, a cloud deployment means that we are going to convert the images in VMs for different IaaS frameworks.

One important feature in our image management design is that we are not simply storing an image but rather focus on the way an image is created through templating. Thus it is possible at any time to regenerate an image based on the template that is used to install the software stack onto a bare operating system. In this way, we can optimize the use of the storage resources. Furthermore, the image repository can maintain specific data that assist in measuring usage and performance. This usage data can be used to purge rarely used images, while they still can be recreated with the use of templating. This will obviously lead to a significant amount of space saving. Moreover, the use of image templating will allow us to automatically generate images for diverse environments including a variety of hypervisors and hardware platforms. In this process, we will include mechanisms to verify that these requirements are reasonable like for example if the required IaaS is compatible with the requested hypervisor. In general, we can employ simple rules such as (a) if we find the image, we just provide it to the user (b)If not, we generate a new image to provide that to the user and store it in the image repository (c) if an image is rarely used it may get purged and we only keep the image generation template.

VIII. Image Compatibility

Currently we support the deployment of images in HPC, OpenStack, Eucalyptus and OpenNebula. We plan to provide support for Nimbus and Amazon as well.

All our images are created in raw format. Thus, when we deploy an image in an infrastructure we only need to do some minor modifications but we do not change its format, see Section VII. These modifications are aimed to allow images take advantages of all features provided by the infrastructure like contextualization or ssh key injection.

Therefore, we could take an image that has been deployed in a particular infrastructure and deploy it in another infrastructure. We should verify that the previous configuration does not conflict with the new one, though.

IX. RAIN - Dynamic Provisioning

Now that we have an elementary way of managing images, we can dynamically provision them onto the resources in bare metal and in virtualized environments while raining them onto our resources. Hence we can

- create customized environments on demand,
- compare different infrastructures, and
- move resources from one infrastructure to another by changing the image they are running plus doing needed changes in the framework.
- ease the system administrator burden for creating deployable images.

As we have provided and demonstrated the basic functionality of rain as part of a scalability experiment, we like to focus in our next tasks on moving resources between the different IaaS and PaaS offerings while integrating this with an advanced reservation system that can be accessed through the queuing system we utilize.

X. Infrastructure Scalability Study

We have performed several tests to study the scalability of the infrastructures installed in our cluster called India. The idea of these tests is to provision as many physical machines (PM) or virtual machines (VM) at the same time as possible. Tests success if all the machines have ssh access. Our results are the time that takes since the request is placed until we have access to all the machines.

For that purpose we have created a CentOS 5 image and deployed it in the different infrastructures: HPC, OpenStack, Eucalyptus and OpenNebula. For that process we have used our image generator and deployment tools. This gives us an identical image to be used in all cases. Therefore, the only difference in the image is the version of the 2.6 kernel/ramdisk used. In HPC we use the ramdisk modified by xCAT, in Eucalyptus we use a XEN kernel and in OpenStack or OpenNebula we use a generic kernel. The total size of the image without compression is 1.5 GB. In the case of netboot it is compress and is around 300MB.

The machines that we are using are Xeon with 8 cores and 24GB of RAM. The network is 1Gbps Ethernet.

Figure 6 shows the results of the performed tests. In the following sections we mention the software used in each infrastructure, the results obtained and the problems we had.

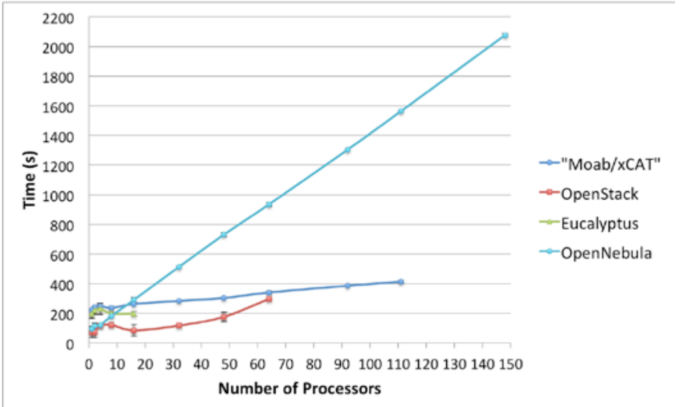


Figure 6: Scalability Experiment of IaaS Frameworks on FutureGrid

A. *HPC (Moab/xCAT)*

In these tests we have used Moab 6.0.3 and xCAT 2.6.9 to dynamically provision the machines. We had a total of 111 machines to be provisioned with the same image. During the tests we did not have any problems and observed very good lineal scalability (see Figure 6). This can be contributed due to good parallel execution behavior that only competes for resources at the time when each retrieves the image that marked for booting. Since the image is compress and relatively small scalability for this setup is preserved. However it is to be noted that provisioning even a single machine takes some time in contrast to its virtualized counterpart. This is due to the process of rebooting the physically machine in case we rain the image on bare-metal.

B. OpenStack

In our experiments, we have used the Cactus version of OpenStack. In order to make an efficient image provisioning, OpenStack caches images in the computes nodes. Thus, it does not need to transfer the image over the network every time it is requested and the instantiation of a VM is faster.

Due to scalability issues by provisioning higher number of VMs, we modified our strategy to submit VM requests in batches of 10 VMs at a time (maximum). This means that in the case of provisioning 32 VMs, we have requested 3 times 10 VMs and once 2 VMs. Without this change we were not able to obtain reliably the requested number f virtual machines within the OpenStack framework deployed in FG. This is a verified bug and the idea stems from the OpenStack team who did the same in their tests [13].

Furthermore, we observed that if the image to be used is not cached in the compute nodes, the scalability of this framework is very limited. In fact, we started to have problems trying to boot 16 VMs, where we got around a 50% of failed tests. Main problems were due to VMs stuck in the launching status (this is the status where the image is copied to the compute node).

On the other hand, once we had the image cached in most of the compute nodes, we were able to boot up to 64 VMs simultaneously. However, this was not an easy task and again we experienced a failure rate of more than a 50%. We observed that VMs got stuck in launching status and other images were in running status but without ssh access. The ssh access problem is mainly related to the fact that OpenStack does not inject the public network configuration inside the VM. Alternatively, it creates bridges and conducts IP forwarding to direct all the traffic toward the private IP. At times we observed that the bridges, the iptables entries or both were not created properly. In particular, the iptables issue is important because we observed that some times OpenStack gets confused and duplicate entries in the iptables using old information that was not properly cleaned up. This erroneous information makes the VMs inaccessible. We observed that this problem is persistent and one needs to manually modify the database to remove the wrong entries or restart OpenStack.

Finally, another known problem of OpenStack cactus is that it is not able to automatically assign public IPs to the VMs, so we have to look into the pool and assign one to each VM manually. This feature has been added in the Diablo version, though. Our intention is to rerun our experiments also with Diablo once it is available on FutureGrid.

C. Eucalyptus

We use the 2.03 version of Eucalyptus, which is the latest OpenSource version. Eucalyptus also caches the images in the compute nodes. As we observed similar issues in Eucalyptus while requesting larger numbers of VMs, the tests were executed in the same way through batched requests like in OpenStack (10 at a time) but with a delay of 6 seconds. This is due to an annoying feature of Eucalyptus that prevents users from executing the same command several times in a short period of time.

The tests with this infrastructure were quite disappointing because we could only get 16 VMs running at the same time, see Figure 6. Even though, the failure rate was very high. Eucalyptus, like OpenStack, configures the public network with IP forwarding and creates the bridges at running time. Thus, this creates problems, like before, and we got similar errors like missing bridges and iptables entries. Additionally, we had problems with the automatic assignment of public IPs to the VMs. This means that some VMs did not get public IPs and therefore they were not accessible for users.

D. OpenNebula

We used OpenNebula version 3.0.0. By default OpenNebula does not cache images in the compute nodes. It supports three basic transfer plugins named nfs, ssh and lvm. NFS has a terrible performance because the VMs are reading the image through the network. SSH was the one that we used because is still easy to configure and has a better performance. The last one seems more difficult to configure, but it should provide the best performance, because it is the one selected by the OpenNebula people to perform their experiments [14, 15]. As we can see in Figure 6, we were able to instantiate 148 VMs at the same time with almost a 100% of success. In fact, we only got one error in the case of 148 VMs. Since our configuration does not use an image cache, it was pretty slow and limited by the network as each image needed to be copied for each VM. This can however be improved by introducing caches as others have proven [16].

XI. CONCLUSIONS

To develop strategies for deployment of IaaS frameworks we have in this paper analyzed some basic functionality and found that the frameworks are sufficient in functionality for many applications. However, we found challenges in our scalability experiments while dynamically provisioning images on them. This was especially evident for Eucalyptus and even OpenStack. As many components are involved in the deployment they are also not that easy to deploy. Tools provided as part of developments such as chef and puppet can simplify deployments especially if they have to be done repeatedly or require modifications to improve scalability. We claim that the environment to conduct an OpenStack experiment with just a handful VMs may look quite different from a deployment that uses many hundreds of servers on which Openstack may be hosted. This is also documented nicely as part of the OpenStack Web pages that recommends more complex service hierarchies in case of larger deployments.

On the other hand, we have seen that OpenNebula is very reliable and easy to deploy. Although in our experiments we found it quite slow due to the lack of cache in the ssh plugin. Nevertheless, we think that this problem is easier to solve than the reliability one found in other frameworks. In fact, after we finished the tests I discovered that one guy modified the SSH plugin to cache the images in the compute nodes.

In all these tests we have manually created the different infrastructures by creating netboot images, configuring networks, adding/removing machines to/from the infrastructures, etc. Therefore it is clear that if we want to do this kind of tests again, we need a tool that does all these things automatically.

We also have to point out that the extension of the rain toolkit by enabling a “move” of resources from on IaaS to another will take some effort, as many services are required to interact. For example, we need to modify the image generation to be able to install more software than the currently support; we need to modify the image deployment to configure the software according to our infrastructure; we need to improve the way that we deal with different kernels and develop a tool that generate different kernels on demand, which is not easy because some software like the hypervisor ones install modules in the kernel. Additionally, the “move” command also needs to be able to disable nodes from one infrastructure and enable them in another, but this has to be done without interfering with running jobs/experiments. This can be an easy task in the case of Moab-PBS because they know what jobs are running and where. However, in the case of Cloud infrastructures it is going to be more complex as we will have to implement some kind of scheduler or queue system to keep track of the running experiments and used hosts.

Finally, one problem that we found during the tests is that we are not ready to netboot images with a Xen kernel because xCAT is not able to generate them. So, we will have to do it manually and this may not be compatible with xCAT. This is needed if we want to dynamically add new hosts to Eucalyptus because the current version only supports this hypervisor or if users need machines with this kernel for their experiments.

On the other hand we have been able to demonstrate that images generated with our tools project cross-platform functionality. Hence we are able to generate images with similar functionality in Eucalyptus, OpenStack, and Nimbus. We can also extend this work by expanding it to for example AWS, or Azure, as well as Nimbus.

Through the ability of rain it will become easier for us to deploy PaaS on the IaaS offerings as we can create “templates” that facilitate their install and potentially their upgrade. Due to this ability it is possible to replicate the environments and introduce reproducible environment.

ACKNOWLEDGEMENTS

We would like to acknowledge the system team for its support and help in the preparation and execution of these tests.

REFERENCES

[1] *Nimbus Project Web Page.* Available: http://www.nimbusproject.org

[2] K. Keahey, I. Foster, T. Freeman, and X. Zhang, "Virtual Workspaces: Achieving Quality of Service and Quality of Life in the Grid," *Scientific Programming Journal,* vol. 13, pp. 265-276, 2005.

[3] *OpenNebula Web Page.* Available: http://www.opennebula.org

[4] I. M. Llorente, R. Moreno-Vozmediano, and R. S. Montero, "Cloud computing for on-demand grid resource provisioning," *Advances in Parallel Computing,* vol. 18, pp. 177-191, 2009.

[5] R. S. Montero, R. Moreno-Vozmediano, and I. M. Llorente, "An Elasticity Model for High Throughput Computing Clusters," *J. Parallel and Distributed Computing,* 2010.

[6] B.Sotomayor, R. S. Montero, I. M. Llorente, and I.Foster, "Virtual Infrastructure Management in Private and Hybrid Clouds," *IEEE Internet Computing,* vol. 13, pp. 14-22, Sep.-Oct 2009 2009.

[7] *OpenStack Web Page.* Available: http://www.openstack.org

[8] *Eucalyptus Web Pages (Open Source).* Available: http://open.eucalyptus.com/

[9] *Eucalyptus Network Configuration 2.0.* Available: http://open.eucalyptus.com/wiki/EucalyptusNetworkConfiguration_v2.0

[10] *Azure.* Available: http://msdn.microsoft.com/en-us/library/dd163896.aspx

[11] *Amazon WS.* Available: http://aws.amazon.com

[12] "Google App Engine."

[13] "Running 200 VM instances on OpenStack."

[14] U. Schwickerath, "CERN Cloud Computing Infrastructure," presented at the ICS Cloud Computing Conference, 2010.

[15] "CERN Scaling up to 16000 VMs."

[16] "Cache optimization in OpenNebula."