[[1]](#footnote-2)

Qualitative Comparison of Multiple Cloud Frameworks

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

*Abstract*— Many cloud infrastructure as a service frameworks exist and users, developers and administrators have to make a decision, which environment is best suited for them. Unfortunately, the comparison of such frameworks is difficult as users may not have access to all of them, or are comparing the performance of such systems on different resources making objective comparisons difficult. Hence, the community benefits from the availability of a testbed on which comparisons between the IaaS frameworks can be conducted. FutureGrid has part of its services offered a number of IaaS including Nimbus, Eucalyptus, OpenStack, but also OpenNebula. One of the important features that FutureGrid provides is not only the comparison between IaaS framework, but also compare them in regards to bare-metal and traditional high-performance computing services. In this paper we outline some of our initial findings by providing such a testbed. As one of our conclusions, we also present our work we started on making access to the various infrastructures on FutureGrid easier.

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

# Introduction

Cloud computing has become an important aspect to deliver infrastructure as a service (IaaS) to users that require a great deal of customization and management of their own software stacks. In addition we observe that users demand further abstraction and expect platforms as a service (PaaS) to be readily available to conduct higher level development efforts. Together IaaS and PaaS can provide potent solutions not only to business users, but also to the educational and scientific computing communities. We observe that in the scientific community we can distinguish a number of user communities.

First, users that demand very view images but want to run them for a long period of time with high-availability demands. This is motivated by exposing services and the delivery of data to large numbers of scientific users utilizing them as part of their research efforts. Such environments are targeting 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 benefit from the delivery of less compute intense services to them.

Second, a handful of scientists that each require a large number of resources to conduct the actual calculations and analyses of data to be exposed to their community. This is the traditional high-performance computing use case.

Third, we find a class of users that is somewhat in-between these two classes with modest but still significant demand on compute resources.

Hence our use cases motivate three classes of infrastructures:

1. Traditional high performance computing (HPC)services offered as part of traditional HPC centers.
2. Hosting services for production like services that cater to a community, including gateways, Web servers and others. Such services may interface with services that run on traditional HPC resources.
3. A testbed to experiment with new usecases to support the development of moderately sized prototypes but also web services and gateways that may be hosted at a later stage of the development on a production machine.

Within this paper we focus on the third class of users. In this paper we will identify the rational for some of the choices that we offer in FutureGrid and identify how to simplify access to an environment that provides so many choices.

The paper is structured as follows. First we will present an overview of FutureGrid and motivate why FutureGrid provides the current services it offers. We will also be able to identify some future directions based on simple usage patterns we observed so far in FutureGrid. One of the main achievements of this paper is to contrast the different IaaS frameworks we offer and project first results of our qualitative comparison. We will provide our thoughts on providing PaaS offerings attractive for our user communities. Next we discuss what implications this multitude of service offerings has for the user community. Based on our qualitative analysis we strive towards providing answers to the following questions:

1. *Which IaaS framework is most suited for me?*
2. *How can I compare these frameworks not just between each other, but also to bare-metal?*

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.

# FutureGrid

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.

In order for users to test out some of the cloud offerings and to identify what kind of applications benefit from clouds FutureGrid offers possibilities to explore a number of IaaS and PaaS frameworks. Currently, FutureGrid provides already a variety of such services. When deciding which services to offer we have based our decision on information we gathered through our web portal. This data was gathered at the time when users applied for a project request via our portal. Each project requestor 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 and . We observed the following:

**Figure 1: 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).**

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 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 Figure 1, as we combined the values for Hadoop and MapReduce while only considering unique entries.
4. We saw recently a surge in requests for OpenStack. It 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.
5. We have seen a increased demand for the support of OpenNebula. It 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.
6. 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 : 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.[[2]](#footnote-3)

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 **Error! Reference source not found.**. 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 servers, while tomorrow I could run OpenStack or Open Nebula on them. This helps in providing a variable testbed and can fulfill scalability experiments that would not be possible by servers that are only dedicated in one IaaS framework.

# 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 frameworks next and outline some major qualitative differences between them.

## Nimbus

The Nimbus project [[1](#_ENREF_1)] is working on two products that they term *“Nimbus Infrastructure”* and *“Nimbus Platform”.*

*Nimbus Infrastructure:* The Nimbus project defines the Nimbus Infrastructureto 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](#_ENREF_1), [2](#_ENREF_2)]

## OpenNebula

OpenNebula [[3](#_ENREF_3), [4](#_ENREF_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](#_ENREF_5)]. OpenNebula can be used to adapt to organizations with changing re­source needs, including addition or failure of physical resources [[6](#_ENREF_6)]. Some essential features to support changing environments includes live migration and snapshots of VMs [[3](#_ENREF_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.

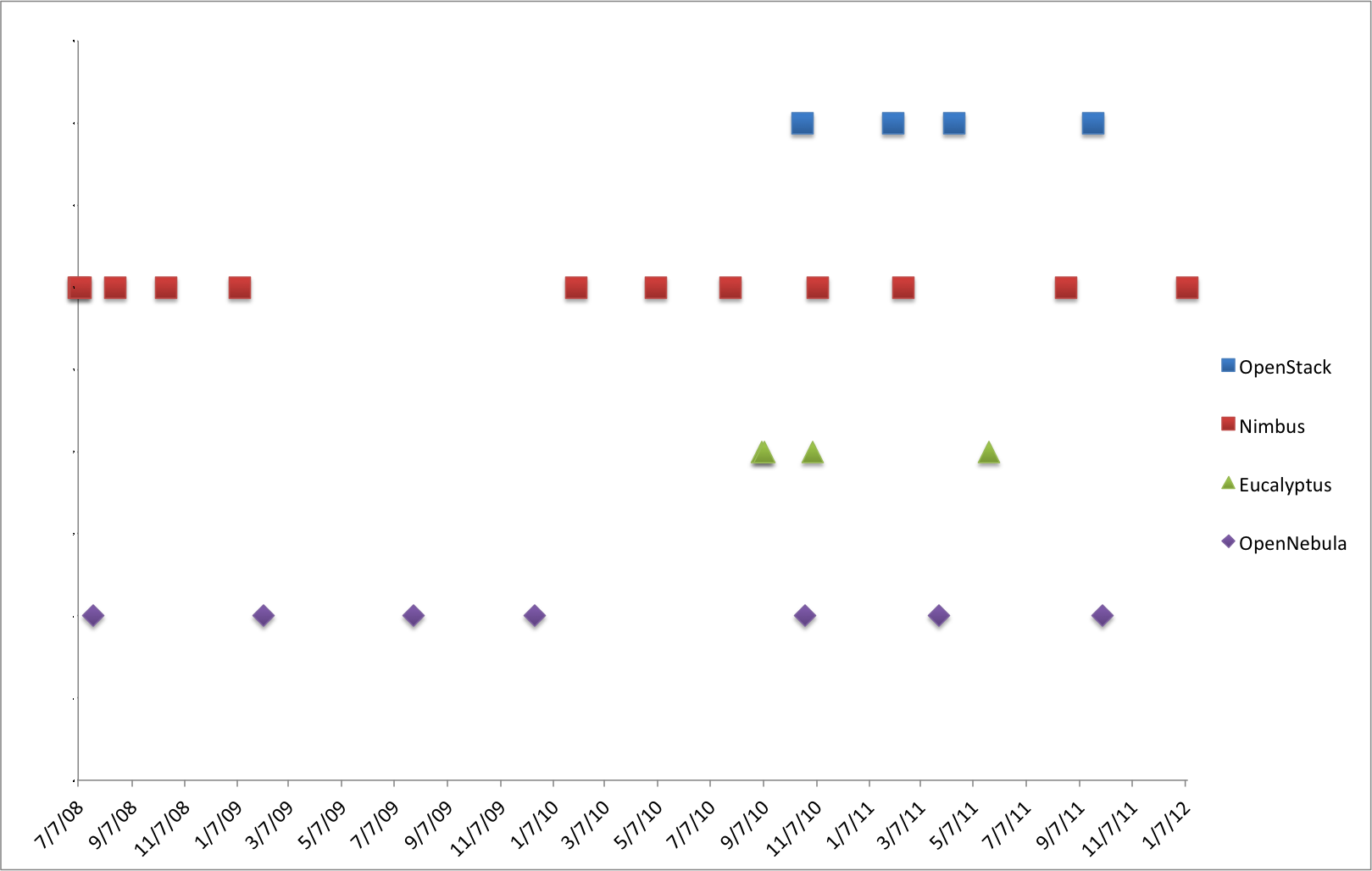


Figure : Release frequency of popular IaaS frameworks (From top to bottom: OpenStack, Nimbus, Eucalyptus, OpenNebula)

## OpenStack

OpenStack [[7](#_ENREF_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.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 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.**

Table : Feature comparison of IaaS frameworks

## Eucalyptus

Eucalyptus [[8](#_ENREF_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.

# Qualitative Feature Comparison of the IaaS Frameworks

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

**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 more sophisticated storage system are provided. In case of OpenStack it is called called Swift, and in case of Eucalyptus it is called Walrus. 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 configured 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](#_ENREF_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 X.509 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.

# Overview of selected PaaS Frameworks

## 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. On FutureGrid we have so far focused on providing Hadoop and Twister to our users as they provide popular choices in the community. Twister is a product developed at IU and used as part of courses taught within the university. This also explains why it is so frequently requested as part of the project registration.

# 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 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.

## Unicore and Genesis II

Additionally, services such as Unicore and Genesis II are available as part of the HPC services.

## Globus

Originally we expected that the demand for Globus would be higher than what we currently see. However, this is explainable due to the following factors: a) Globus demand has been shrinking b) many new projects investigate cloud technologies instead of using Grid toolkits c) Grid services have been offered in stable form as part of OSG and XSEDE and thus the demand within a testbed is minimal. Projects that requested Globus include the SAGA team to investigate interoperability, the EMI project, and the Pegasus Project. Only the later has so far requested a real requirement of hosting Globus on bare metal in a permanent fashion. The other projects have indicated that their needs can be fulfilled by *raining*  Globus onto a server based on images that are managed by these projects. Furthermore, we observe that the Globus project has moved over the last year to deliver package based installs that make such deployments easy. We had only one additional user that requested not the installation of Globus, but the installation of GridFTP to more easily move files along. While analyzing this request, it could easily be satisfied while using the CoG Kit and abstracting the need for moving data away from a protocol towards the function of provide me with a simple and fast way to move my data. Interoperability projects would be entirely served by a virtual Globus installation on images via rain (see Section VII). In many of these cases even bare-metal access is not needed. In fact such use cases benefit from the ability to start up many Globus services to as part of the development of advanced workflows that are researched by the community.

At this time we are not supporting any other PaaS offerings such as messaging queues or hosted databases. However, we adapt our strategies based on evaluating user requests with available support capabilities.

# RAINING

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. As part of this process, we allow the generation of abstract images generation and universal image registration with the various infrastructures including Nimbus, Eucalyptus, OpenNebula, OpenStack, but also bare-metal via the HPC services. This complex process is described in more detail in [17]. It is one of the unique features about FutureGrid to provide an essential component to make comparisons between the different infrastructures more easily possible. Our toolkit rain is tasked with simplifying the full life cycle of the image management process in FG. It involves the process of creating, customizing, storing, sharing and deploying images for different FG environments. 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.

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,
* move resources from one infrastructure to another by changing the image they are running plus doing needed changes in the framework, and
* ease the system administrator burden for creating deployable images.

This basic functionality has been tested as part of a scalability experiment motivated by our usecase introduced in the introduction. A user likes to start many virtual machines and compare the performance of this task between the different frameworks.

# 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 4 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.

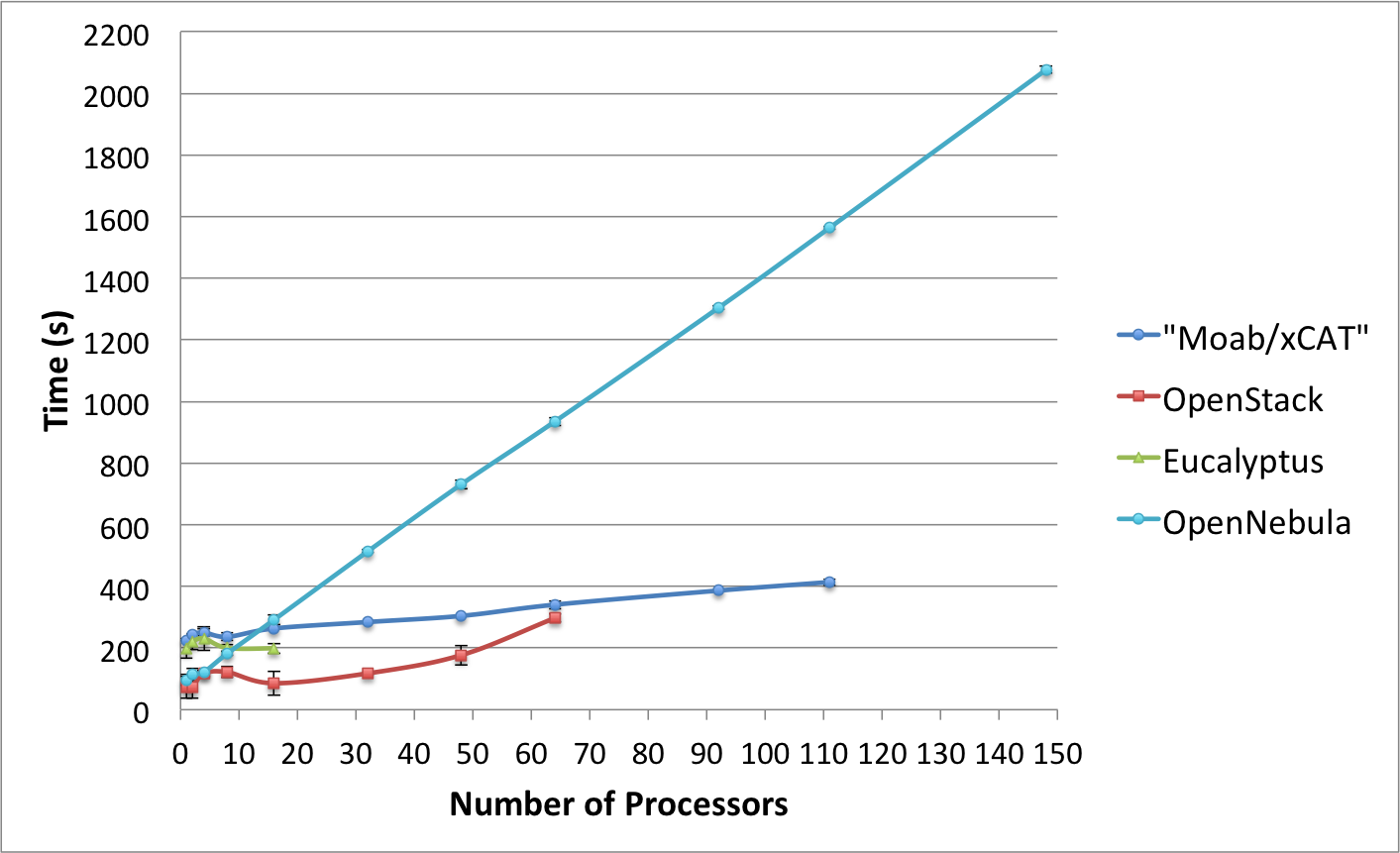


Figure : Scalability Experiment of IaaS Frameworks on FutureGrid

## 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.

## 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](#_ENREF_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.

## 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 4. 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.

## 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](#_ENREF_14), [15](#_ENREF_15)]. As we can see in Figure 4, 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](#_ENREF_16)].

# Conclusions

This paper provides evidence that offering not one but many IaaS frameworks is needed to allow users to answer the question which IaaS framework is right for them. We have to be aware that the users requirements may be very different from each other and the answer to this question can not be universally cast. Within this paper we first tried to give a motivation which IaaS frameworks to focus on and have based on user feedback and community monitoring identified that it is best to offer all of them at this time.

We also identified a simple question posed by many our users: Which infrastructure framework should I use under the assumption I have to start many instances. This is naturally motivated by our close ties to the HPC community and researchers utilizing parallel services and programming methodologies due to their highly complex problems requiring it. However, such users tend not to be experts in IaaS farmeworks and a tool is needed that allows the comparison of their usecases in the different frameworks. In this paper we have demonstrated that we have achieved a major milestone towards this goal and proven that our efforts can lead to significant coomparisons between deployed frameworks. This also includes the ability to identify a set of benchmarks for such infrastructures in order to further identify which framework is best suited for a particular application. Our test case must be part of such a benchmark.

Concretely, we found challenges in our scalability experiments while raining 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 is quite slow due to the lack of cache in the ssh plugin. Nevertheless, we think that this problem is easy to solve. In fact, after we finished the tests we also found othe community members having the same problem. A plugin to the caching problem is by now available that can significantly improve managing images across many nodes in OpenNebula.

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 rest of the FG 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."

[17] Abstract Image Management and Universal Image Registration for Cloud and HPC Infrastructures Javier Diaz, Gregor von Laszewski, Fugang Wang and Geoffrey Fox Pervasive Technology Institute, Indiana University, Paper Draft,

<http://cyberaide.googlecode.com/svn/trunk/papers/a-draft/draft-11-imagemanagement/draft-11-imagemanagement.pdf>

[18] P. Sempolinski and D. Thain. A Comparison and Critique of Eucalyptus, OpenNebula and Nimbus. IEEE 2nd Int. Conf. on Cloud Computing Technology and Science (CloudCom), 2010.

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. [↑](#footnote-ref-2)
2. 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. [↑](#footnote-ref-3)