

Accessing Multiple Clouds with Cloudmesh

Gregor von Laszewski*
School of Informatics and
Computing, Indiana University
919 E. 10th Street
Bloomington IN 47408, U.S.A.
laszewski@gmail.com

Fugang Wang
School of Informatics and
Computing, Indiana University
919 E. 10th Street
Bloomington IN 47408, U.S.A.

Hyungro Lee
School of Informatics and
Computing, Indiana University
919 E. 10th Street
Bloomington IN 47408, U.S.A.

Heng Chen
School of Informatics and
Computing, Indiana University
919 E. 10th Street
Bloomington IN 47408, U.S.A.

Geoffrey C. Fox
School of Informatics and
Computing, Indiana University
919 E. 10th Street
Bloomington IN 47408, U.S.A.

ABSTRACT

We present the design of a toolkit that can be employed by users and administrators to manage virtual machines on multi-cloud environments. It can be run by individual users or offered as a service to a shared user community. We have practically demonstrated its use as part of a FutureGrid service, allowing users of FutureGrid to utilize such a service. Furthermore, we discuss implications and solutions for a unified metrics system assisting the users to find and utilize resources appropriate for their applications. Lastly, we discuss how to move such a multi-cloud environment forward by integrating clouds managed by the community or are offered as public clouds. This includes the introduction of a mutual trust agreement on a user and project basis. We have developed a number of components that support the creation of such a multi-cloud environment. This system is known as Cloudmesh and has been used in practice to achieve virtual machine management in multiple clouds. An important distinguishing factor of Cloudmesh is that it also allows the use of bare metal provisioning for supporting service providers and authorized users, offering services beyond those available by typical clouds.

Categories and Subject Descriptors

C.2.4 [Distributed Systems]: Cloud Computing

General Terms

Computer-Communication Networks

*Corresponding author.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
BigSystem 2014, June 23, 2014, Vancouver, BC, Canada.
Copyright 2014 ACM 978-1-4503-2909-5/14/06 ...\$15.00.
<http://dx.doi.org/10.1145/2609441.2609638>.

Keywords

Cloud, multi-cloud, federated clouds, FutureGrid, Cloudmesh

1. INTRODUCTION

Cloud computing has become an integral factor for managing infrastructure by research organizations and industry. Users and organizations are faced with a variety of solutions that may support their needs. Such offerings include a variety of Infrastructure as a Service (IaaS) frameworks. The choice between them provides a significant risk of investment and has to be conducted carefully. The community has the choice to either use public clouds or set up their own private clouds. Public clouds are offered by large providers such as Amazon, Microsoft, Google, Rackspace, HP, and others. Private clouds are set up by internal Information Technology (IT) departments and made available as part of the general IT infrastructure. Two aspects resulting from this availability bear consideration. First, which of the many Infrastructures should be chosen? Second, can the multitude of IaaS service offerings be leveraged and instead of choosing one, can a service be offered that utilizes multiple services? Furthermore, we have to consider that the IaaS frameworks to deploy private clouds are evolving and that extensions may be needed that are not yet offered by the frameworks to adequately address the requirements posed by advanced IT service infrastructures of research organizations.

In this paper, we will present the design and implementation of a toolkit that addresses some of the points raised here. The paper is structured as follows. First, we discuss the important terminology used in this paper (Section 1.1). Next, we analyze the requirements in more detail (Section 3) and summarize some related work (Section 2). Then, we explore the requirements that motivate our design (Section 4) and implementation (Section ??). We focus on two aspects of the Cloudmesh features, namely dynamic provisioning/rain. Last, we present our conclusion (Section 6).

1.1 Terminology

We will be using the following definitions throughout the paper:

Public cloud: a service provider makes resources available to users over the public internet. This includes compute, storage, and applications. FutureGrid offers a public cloud to its users.

Private cloud: access to services may have additional restrictions. Restrictions could include a limited set of authorized users to the services offered or possible restrictions regarding exposing services on the public internet. FutureGrid offers the ability to set up private clouds for special projects. Examples include modified OpenStack deployments or reserved resources for classes.

Hybrid cloud: a combination of public and private clouds.

Multi-cloud: access to a number of different clouds that may even use different IaaS or PaaS offerings.

HPC service: a cloud service that allows the ability to run high performance computing jobs, for example on a compute cluster offering MPI.

Provisioning: a set of instructions to install the operating system, data and software to enable access to it.

Rain: an advanced set of instructions that not only provisions the operating system, but allows the deployment and configuration of useful and complex services to be run on one or multiple machines in order to provide a service utilizing potentially distributed resources or services. It also contains the ability to re-provision servers and services, that is, services may be suspended and the resources used to run the service may be used by other services.

Provider consortium: is a (virtual) organization that integrates resources from multiple providers. We also can refer to such a consortium as a multi-cloud Grid.

2. RELATED WORK

It is not possible to provide an extensive overview of related research due to space limitations of this paper. Instead, we will provide examples of activities that address similar although not the same issues as our work.

Phantom [6] is a tool that monitors the health of resources and automatically provisions and configures new ones based on demand. It is designed to automatically (a) provision new resources to counteract failures or (b) react to increasing demand, reducing constant attention and repetitive tasks that can be automated.

What makes our effort different, is that our framework is focused not only on the user initiated federation of resources, the access to the native protocols that are not only the reliance on EC2 EC2 protocol, as well as the concept of access to baremetal provisioning. Furthermore, our framework includes the ability to provide a holistic cloud usage service that can be used to develop holistic scheduling algorithms for better utilization.

RightScale [8] enables users to manage multi-cloud infrastructure by migrating workloads between private clouds and public clouds. RightScale interfaces with a wide variety of clouds including Amazon Web Services (AWS), Rackspace Cloud, Windows Azure, and Google Compute Engine. It also offers a cloud cost estimator, allowing customers to assess expenses they are charged by comparing their workload on various cloud providers.

Our effort is different because it is an open source toolkit and allows the deployment not only as a hosted service managed

by one entity, but also allows the deployment by a provider, provider consortium, or even the user.

Other research efforts include theoretical definitions for cloud federation [7] or are relying on a single IaaS, such as efforts planned for future versions of OpenStack. Standard efforts provide an interesting approach to multi-cloud interoperability, but at the same time may hinder the innovation brought forward by individual IaaS offerings. We see such an encumbrance with the difference between the OpenStack and EC2 API's, which leads to limitations in the functionality of Phantom.

2.1 FutureGrid

Many of the requirements for this project originate from FutureGrid. FutureGrid [12, 3] is a project to develop a high performance grid testbed that will allow scientists to collaboratively develop and test innovative approaches to parallel, grid, and cloud computing. It provides infrastructure and advanced services that are documented in more detail in [11]. FutureGrid users can deploy their own hardware and software configurations on a public/private cloud, and run their experiments. FutureGrid provides an advanced framework to manage user and project affiliation and propagates this information to a variety of subsystems constituting the FutureGrid service infrastructure. This includes operational services to deal with authentication, authorization and accounting. In particular we have developed a unique metric framework [13] that allows us to create usage reports from our entire IaaS frameworks and is presented in more detail later. Repeatable experiments can be created with a number of tools including Pegasus, Precip and Cloudmesh. Provisioning of services and images can be conducted by Rain [1, 2]. One of the main features of FutureGrid is to offer its users a variety of IaaS frameworks [10] including OpenStack, Eucalyptus, and Nimbus. Based on our experience with FutureGrid over the last couple of years, it is advantageous to offer a mixed operation model. This includes a standard production cloud that operates on-demand, but also a set of cloud instances that can be reserved for a particular project.

3. REQUIREMENTS

We list here a subset of requirements that are to be addressed by the design and implementation of our toolkit. This includes (a) provide virtual machine management in a multi-cloud environment while using (a.1) FutureGrid resources, (a.2) external clouds from fresearch partners, (a.3) public clouds; (b) provide multi-cloud services controlled by the user; (c) provide multi-cloud services controlled by the provider; (d) enable deployment of the service by users or providers; (e) enable raining (e.1) of operating systems (baremetal provisioning), (e.2) services, (e.3) platforms, (e.4) IaaS; (f) monitoring infrastructure across a multi-cloud environment; (g) provide multiple interfaces such as (g.1) command line, (g.2) command shell, (g.3) REST, (g.4) Python API; (h) deliver an (h.1) open source (h.2) extensible (h.3) easy deployable, (h.4) documented toolkit.

As open source project much of the presented material in the next sections can be found in its online Web page [5].

4. DESIGN

Our initial design is addressing the requirements listed in Section 3 and will provide a tightly integrated software in-

frastructure toolkit addressing the need to deliver a software-defined system encompassing virtualized and bare-metal infrastructure, networks, application, systems and platform software with a unifying goal of providing Cloud Testbeds as a Service (CTaaS). This system is termed Cloudmesh to symbolize

1. the creation of a tightly integrated mesh of serviceto sys targeting multiple IaaS frameworks (Requirement (a)),
2. the ability to federate a number of resources from academia and industry. This includes existing FutureGrid infrastructure, Amazon Web Services, Azure, HP Cloud, Karlsruhe using not only one IaaS framework but various (Requirements (a), (b), (c)),
3. the creation of an environment in which it becomes more easy to experiment with platforms and software services while assisting to deploy them more easily (Requirement (e)).
4. the exposure of information to guide the efficient utilization (Requirement (f))

In addition to virtual resources, Cloudmesh exposes baremetal provisioning to users. Services will be available through command line, API, and Web interfaces (Requirement (g)).

Due to its integrated services Cloudmesh provides the ability to be an onramp for other clouds. It also provides information services to various system level sensors to give access to sensor and utilization data. Internally, it can be used to optimize the system usage. The provisioning experience from FutureGrid has taught us that we need to provide the creation of new clouds, the repartitioning of resources between services (cloud shifting), and the integration of external cloud resources in case of over provisioning (cloud bursting). As we deal with many IaaS frameworks, we need an abstraction layer on top of the IaaS framework. Experiment management is conducted with workflows controlled in shells [9], Python/iPython, as well as systems such as OpenStack’s Heat. Accounting is supported through additional services such as user management and charge rate management. Not all features are yet implemented. Figure shows the main functionality that we target at this time to implement.

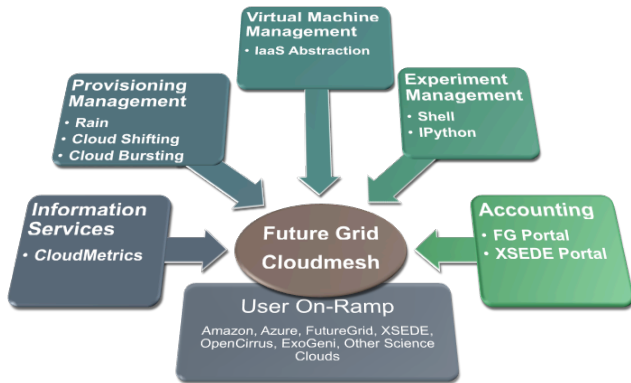


Figure 1: Cloudmesh Functionality.

To implement the Cloudmesh functionality, we have devised a layered architecture to gradually provide services in support of the requirements identified for our toolkit.

The architecture is depicted in Figure 2 and contains three main layers: the resource layer, the provisioning and execution layer, and the management framework layer.

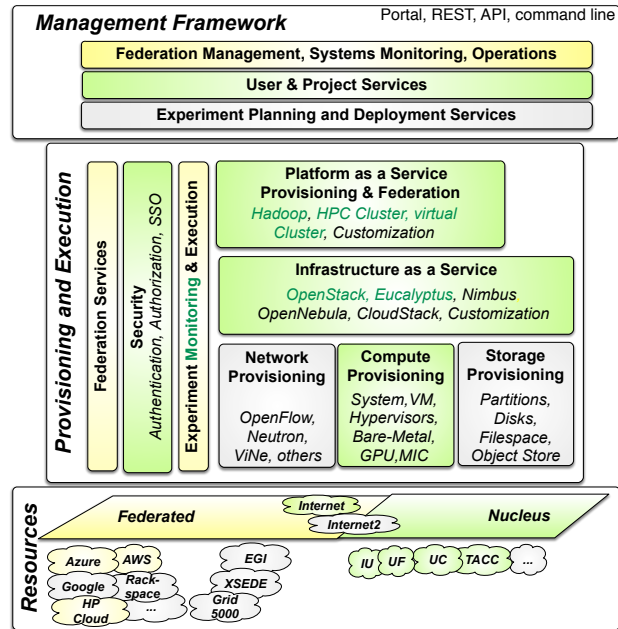


Figure 2: Cloudmesh Architecture.

4.1 Management Framework Layer

User and Project Services. FutureGrid user and project services simplify the application processes needed to obtain user accounts and projects. We have demonstrated in FutureGrid the ability to create accounts in a very short time, including vetting projects and users – allowing fast turn-around times for the majority of FutureGrid projects with an initial startup allocation. Cloudmesh reuses this infrastructure and also allows users to manage proxy accounts to federate to other IaaS services to provide an easy interface to integrate them.

Accounting and App Store. To lower the barrier of entry, Cloudmesh will be providing a shopping cart, which will allow checking out of predefined repeatable experiment templates. A cost is associated with an experiment making it possible to engage in careful planning and to save time by reusing previous experiments. Additionally, the Cloudmesh App Store may function as a clearing-house of images, image templates, services offered and provisioning templates. Users may package complex deployment descriptions in an easy parameter/form-based interface and other users may be able to replicate the specified setup. Due to our advanced Cloudmesh Metrics framework we are in the position to further develop an integrated accounting framework allowing a usage cost model for users and management to identify the real impact of an experiment on resources. This will help avoid overprovisioning and inefficient resource usage. The cost model will be based not only on number of core hours used, but also the capabilities of the resource, the time, and special support it takes to set up the experiment. We will expand upon the metrics framework of FutureGrid that allows measuring of VM and HPC usage and associate this

with cost models. Benchmarks will be used to normalize the charge models.

4.2 Provisioning and Execution Layer

Baremetal Provisioning. We have a broad vision of resource integration offering different levels of control from bare metal to use of a portion of a resource. As part of our efforts we originally developed a provisioning framework based on XCAT. However due to limitations and significant changes between versions we are currently replacing it with a more general framework that allows the utilization of different bare metal provisioners. At this time we have provided an interface for cobbler and are also targeting an interface to OpenStack Ironic.

Virtual Machine Provisioning. Cloudmesh provides an abstraction layer to allow the integration of virtual machine management APIs based on the native IaaS service protocols. This helps in exposing features that are otherwise not accessible when quasi protocol standards such as EC2 are used on non-AWS IaaS frameworks. It also prevents limitations that exist in current implementations, such as libcloud to use OpenStack.

Network Provisioning. Likewise, we must utilize networks offering various levels of control, from standard IP connectivity to completely configurable SDNs as novel cloud architectures will almost certainly leverage NaaS and SDN alongside system software and middleware. FutureGrid resources will make use of SDN using OpenFlow whenever possible though the same level of networking control will not be available in every location.

Storage Provisioning. As we have access to baremetal provisioning storage, provisioning becomes a possibility while keeping partitions between deployments and experiments. However, we will have to expand upon making storage provisioning accessible to the users.

Platform, IaaS, and Federated Provisioning. One of the significant features of the design of Cloudmesh is that it can not only provision the operating system, but that through additional services it can also provision platforms, IaaS, and even federated services. This is achieved by the integration of cloudmesh shell scripting, or the utilization of DevOps frameworks such as Chef or Puppet.

Furthermore, we have already demonstrated via the Rain tool in FutureGrid that it is possible to *shift* resources allocations between services such as HPC and OpenStack or Eucalyptus. We are currently expanding upon this idea and developing intuitive user interfaces as part of Cloudmesh that assist administrators and users through role and project based authentication to move resources from one service to another (see Figure 3).

4.3 Resource Layer

To integrate IaaS frameworks Cloudmesh offers two distinct services: (a) a federated IaaS frameworks hosted on FutureGrid, (b) the availability of a service that is hosted on FutureGrid, allowing the integration of IaaS frameworks through user credentials either registered by the users or automatically obtained from our distributed user directory.

For (b) several toolkits exist to create user-based federations, including our own abstraction level which supports interoperability via libcloud, but more importantly it directly

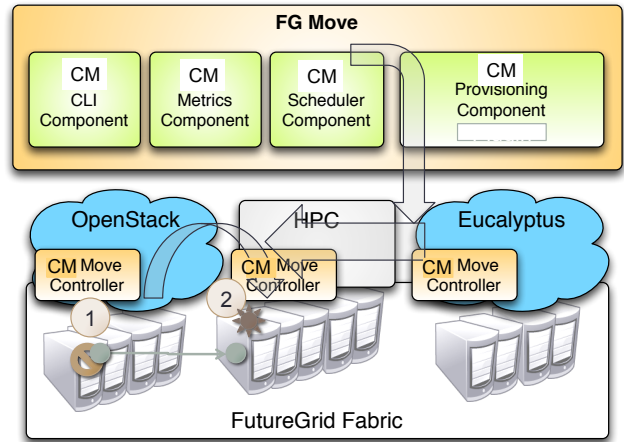


Figure 3: Shifting resources makes it possible to offer flexibility in the service distribution in case of over or underprovisioning.

supports the native OpenStack protocol and overcomes limitations of the EC2 protocol and the libcloud compatibility layer. Plugins that we currently develop will enable access to clouds via firewall penetration, abstraction layers for clouds with few public IP addresses and integration with new services such as OpenStack Heat. We successfully federated resources from Azure, AWS, the HP cloud, Karlsruhe Institute of Technology Cloud, and four FutureGrid clouds using various versions of OpenStack and Eucalyptus. The same will be done for OpenCirrus resources at GeorgiaTech and CMU. Additional management flexibility will be introduced through automatic cloud-bursting and shifting services. While cloud bursting will locate empty resources in other clouds, cloud shifting will identify unused services and resources, shut them down and provision them with services that are requested by the users. We have demonstrated this concept in 2012 moving resources for more than 100 users to services that were needed based on class schedules. A reservation system will be used to allow for reserved creation of such environments, along with improvements of automation of cloud-shifting.

4.4 Monitoring

Cloudmesh must be able to access empirical data about the properties and performance of the underlying infrastructure beyond what is available from commercial cloud environments. The component of Cloudmesh accomplishing this is called *Cloud Metrics*.

Experience with FutureGrid has provided greater understanding about resource allocation and utilization. We have learned that it is essential to provide users and administrators with a holistic view of the infrastructure in order to guide better utilization overall and on an individual basis. This is crucial in cloud deployments such as FutureGrid, which supports more than 380 projects and 2300 users (as of April 2014). Due to such a large user base, resources could become over-provisioned or not properly utilized by the users. Among the many services that FutureGrid offers, we have particularly focused on IaaS including OpenStack, Eucalyptus, Nimbus, as well as batch systems, to offer high performance computing capabilities. However, for this pa-

per we will restrict our discussion to the IaaS based monitoring components. Other HPC related activities in regards to monitoring and metrics are discussed in [4].

When FutureGrid initially started, the existing IaaS frameworks such as Eucalyptus, Nimbus, and OpenStack did not provide adequate support for monitoring resource usage. Furthermore, a service with sufficient monitoring capabilities across heterogeneous cloud IaaS frameworks was not available. Hence, it was difficult to assess user utilization in a holistic fashion. Additionally, we found that some IaaS frameworks such as Nimbus, lacks support for project allocations, which is a must have feature to support project managed allocations as is the case of almost every modern shared datacenter. To overcome these missing features and service offerings, we developed a *federated cloud metric service* that aggregates the information from distributed clusters and a variety of heterogeneous IaaS services, such as OpenStack, Eucalyptus, and Nimbus. We have named this service *Cloudmesh Metrics*.

The main components of *Cloudmesh Metrics* enable (a) the measurement of the resource allocation across several IaaS platforms, (b) the generation of data in regards to utilization, (c) the comparison of data via definable metrics to mine the usage statistics, (d) the display of the information through a convenient user interface, (e) the availability of a simple command line interface and shell language, and (f) the automatic creation of resource reports in printed format for arbitrary time periods.

The services offered by Cloudmesh Metrics support requirements from a variety of user communities. This includes individual users and users as part of projects (Section 4.4.1), as well as administrative users (Section 4.4.2).

4.4.1 User- and Project-based Metrics Services

In order for users to use a variety of clouds, it is important for them to monitor and compare their resource utilization. In case the usage is organized as a project, the project related information needs to be exposed while being able to clearly distinguish between different projects. Therefore, we need to support an overall project view. Thus, the requirement exists to present the data to the user based on individual user utilization, group utilization, or even experiment utilization, where a particular experiment is analyzed instead of just looking at the total utilization.

4.4.2 Resource Provider-based Metrics Services

A resource provider need to have access to a holistic view of the variety of metrics across the different clouds that build the multi-cloud environment as part of a provider consortium. Summary information may be customized based on the requirements by individual users, project leads, resource providers, site managers, and funding agencies. This information is typically restricted to the actual *resources* for which administrative access exists in order to provide a holistic set of metrics.

4.4.3 Metric Access for Multi-cloud Environments

Due to the different governance models between a private cloud managed as part of a provider consortium, and the integration of resources, use must be based on an access integration policy. In order to devise such a policy, we need to be aware of the hierarchical access management employed in clouds as depicted in Figure 4.

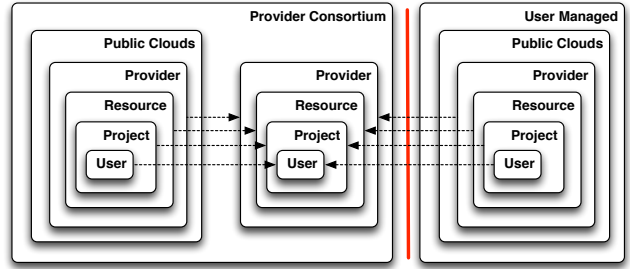


Figure 4: An example of a policy to establish a access public clouds as part of provider and user managed multi-clouds.

Integration into a Provider Consortium Managed Multi-Cloud.

In the hierarchy, we distinguish users \Rightarrow project with many users \Rightarrow a resource serving many projects \Rightarrow a provider having many resources \Rightarrow and a provider consortium with many providers.

The provider consortium has a multitude of possibilities to extend their resource offerings to its users while integrating public clouds. This could include replicating projects on public clouds, or assigning a particular user access to an account that integrates resources in a public cloud. In case multiple clouds are offered, this integration could be replicated for them. Hence, it would be possible for a provider consortium not only to provide private clouds as part of a multi-cloud environment, but also public cloud offerings allowing access to these public clouds through a provider consortium managed project or users on the public clouds. This is of particular interest if we intend to gain financial advantages through volume discounts that would otherwise not be available to the users. In Figure 4 we show one example for such an integration policy where we simply map the existing projects and users of the consortium to projects and users of a public cloud. An applicable instance where such integration is easy to accomplish is the HP Cloud environment that uses OpenStack as its IaaS framework. Due to OpenStack's well-documented interfaces, it is possible to replicate the user and project information and provide detailed charges to the users on projects in case the HP OpenStack cloud would be used. Naturally, it would be possible to devise other integration policies such as restricting access to only approved projects, or provide limited access into different levels of the hierarchy.

Integration into a User Managed Multi-Cloud.

Based on our experience with FutureGrid, we must also be aware that users may have their own accounts and access to other clouds, thereby needing to integrate them into a multi-cloud environment. The user decides whether the metric information to such clouds is forwarded to the multi-cloud environment offered as part of a Provider Consortium. However, in most cases the user will not share this information. Hence, the metric system ideally should be able to allow users to integrate their own information into such a metric system in order for the user to gain a more accurate picture about their own cloud usage not just in the consortium, but also in the public clouds. We represent this clear division in Figure 4. Explicit access policies must be de-

fined to allow the users or projects to access the information provided by the public clouds.

Cloudmesh Metric Architecture.

The Cloudmesh metric architecture is based on the integration of an authorized REST service, that utilizes a simple abstraction layer to interface with the various cloud services to obtain needed information gathered under authorization constraints. The data will be hosted in a NOSQL database to allow mining of the data in map/reduce frameworks. Data can be ingested either directly through the database via the API, or through REST calls that are mitigated through message queues with AMPQ. Adapters can be written to integrate new information providers for other clouds. Policies can be used to limit the amount of information presented to other users or projects.

4.4.4 Cloudmesh Metric Service for FutureGrid

To work towards the goal of a metric system for multi-cloud environments, we started by basing our initial development efforts on the extension of the cloud metrics services that have been developed by FutureGrid for a multi-cloud environment for resource providers within FutureGrid. Here, we have limited the services to a provider consortium that offers information of clouds directly managed by the consortium. This includes OpenStack, Eucalyptus, and Nimbus clouds on various resources. The information access policy for using the resources is public, as this is most suitable to the goals of FutureGrid as a public testbed.

Data Collector and Metric service.

One of the fundamental services needed is a data collector. It collects relevant data from a variety of sources including resource databases, log files, and data reporters. Hence, to integrate new cloud services into the data collectors we have to define a data model, as well as data sources that populate the data model with data. Currently, data collectors are available for OpenStack, Eucalyptus, and Nimbus but not limited to these platforms. Dependent on the IaaS framework they obtain the data from different sources such as log files or databases as listed in Table 1. Beneficial information to be collected includes detailed information about the virtual machines (VM), the users and/or projects starting them, memory usage over the lifetime of the VMs, errors associated with VMs during runtime, or at startup. One of the issues to be addressed is, if such data should be directly accessed in the production environment offered by the IaaS framework. Practical experience with FutureGrid has shown that the analysis of the data poses a significant amount of stress on the originating resource, making it impractical to offer a detailed report and metric system on the original data sources. Hence, it is important that we replicate it when the information we request is involved in a detailed analysis. For some smaller scale queries, as the one posed by most users, direct access is sufficient and desirable for the live view of the system in order to provide information about how many VMs are currently running, on which system and by whom.

Metric Analyzer.

The data collected provides the opportunity to analyze it for specific needs in a repeated fashion or provide filters and services for further specialized analysis. The FutureGrid metric framework therefore provides a metric analyzer com-

Table 1: Measurement of IaaS on FutureGrid

	Nimbus	OpenStack	Eucalyptus
Documentation of the Data Sources			
	X	✓	✓
Data Sources			
	sqlite3	MySQL	Log Files
Metrics			
vCPU core	✓	✓	✓
memory	✓	✓	✓
disk	✓	✓	✓
instance type	X	✓	✓
host	✓	✓	✓
Account Management Features			
Users	✓	✓	✓
Projects	X	✓	✓
Cluster			
Alamo	✓	✓	X
Foxtrot	✓	X	X
Hotel	✓	✓	X
India	X	✓	✓
Sierra	X	✓	✓

ponent with a convenient interface for analyzing the data not only on an automated fashion, but also interactively through a simple metrics analyzer shell. Information of interest includes yearly, monthly, and/or weekly usage information by user, project, resource, provider, and the agglomerated information. Our scripting environment provides this information and is run at predefined intervals or upon request. In the future, we will be enhancing the service to allow users to schedule queries to conduct specific analysis. To avoid repeated analysis, metric result caching is conducted. Thus, if a query has been executed in the past the result is cached and returned without reanalysis (if not forced). To more easily facilitate fast and distributed calculation of the results by multiple users, we will base future versions of the Cloudmetric system on NoSQL database technologies.

Metric Interface.

Early on we recognized that the access of information and the metrics must be provided through a variety of interfaces. This includes command shells, programming API's, REST interfaces, graphical user interfaces, and printable reports.

Interactive Command Shell. To simplify the interactive use, we have developed a python command shell called CMD3 that allows the dynamic load of additional commands, thus making it ideal to define new analytic methods on the fly if they are not provided by the original toolkit.


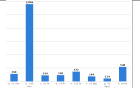


REST API. We are currently building access through a convenient REST API to allow easy access from Web frameworks, but also integration from arbitrary programming languages

Programming API. We have provided a robust API interface in python to access the basic analytical functions useful for many users and reused by the interactive command shell and the REST service.

Graphical Representation and Printable Reports.

Using our basic API and command shell, we have integrated them into the Python sphinx framework to expose the metric data in a convenient form and present the data online via charts or in a PDF report. As sphinx offers the

Table 2: Metric visualization with graphs

Example	Description
	Detailed display of virtual machine information including number of virtual machines, user count, memory utilization, disk utilization, project lead, etc.
	Summary information for periods to display aggregates of the metrics gathered by the system, such as number of VMs by month for a user, project, or resource.
	Alternate representation of aggregated information in pie charts.
	Alternate representation of agglomerated information in a table (exportable as csv or json).

export of data reports in PDF, we leverage this framework and do not have to develop a separate framework for it. The sphinx framework service is currently enhanced, allowing customizable interactive queries to the metric and data sources. The data can be represented visually in various chart forms such as bar graphs, line charts, or pie charts. A template for generating a quarterly and yearly report of the data exists making adaptation to additional resources or other provider consortia easy. Furthermore, the data can be exported in a variety of formats such as JSON or CSV making it possible to use other tools, such as excel for data post processing. In Table 2 we are including a limited number of examples to demonstrate the various data representations of the Cloudmetric system that are exposed to the users.

4.5 Graphical User Interface

Despite the fact that Cloudmesh was originally a quite sophisticated command shell and command line tool, we recently dedicated more time in exposing this functionality through a convenient Web interface. Some more popular views if this interface are depicted in Figure 6 hinting on how easy it is with a single button to create multiple VMs across a variety of IaaS. Notably, this not only includes resources at IU but also at external locations. Pushing this easy management in a more sophisticated experience for the user while associating one-click deployments that include the ability to deploy virtual clusters, Hadoop environments, and other more elaborate setups we provide an early prototype screenshot in Figure 7.

5. STATUS AND FUTURE WORK

At this time we have developed a first version of Cloudmesh and focussed on the development of three of its components. This includes virtual machine management in multi-clouds, cloud metrics in multi-clouds, and bare metal provisioning. The next steps will include the development of other components we discussed in our design. Cloudmesh has been successfully used in FutureGrid. A GUI and a Cloudmesh shell is available for easy usage by users. It has been used by users while deploying it on their local machines, but it also has been demonstrated as a hosted service. A REST interface to the management functionality is under development. Cloudmesh is an open source project. It uses python and Javascript. Although the current Web related func-

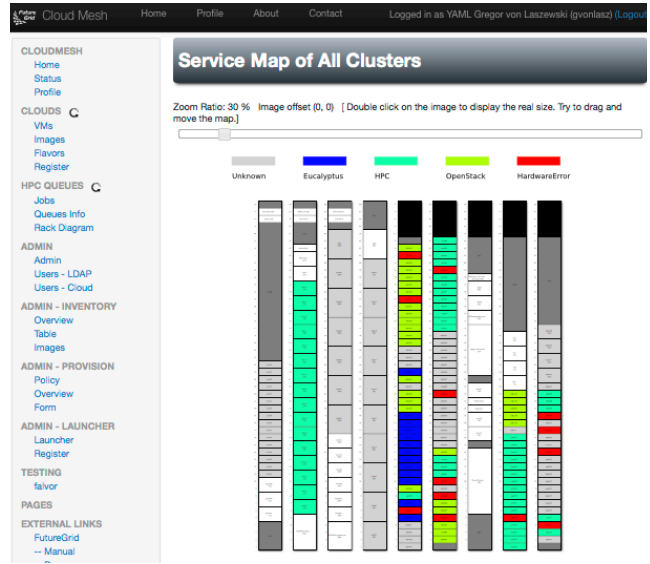


Figure 5: Monitoring the service distribution in FutureGrid Clusters with Cloudmesh.

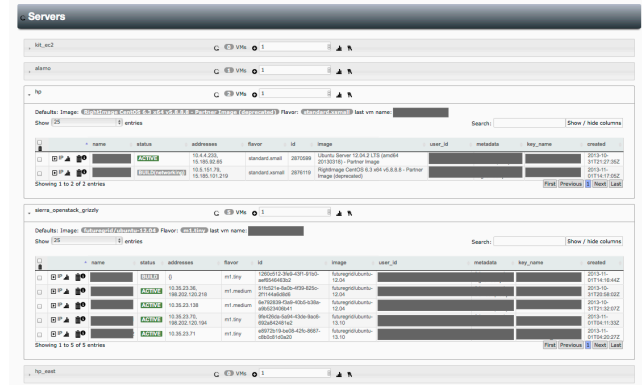


Figure 6: Screenshot demonstrating how easy it is to manage multiple VMs across various clouds.

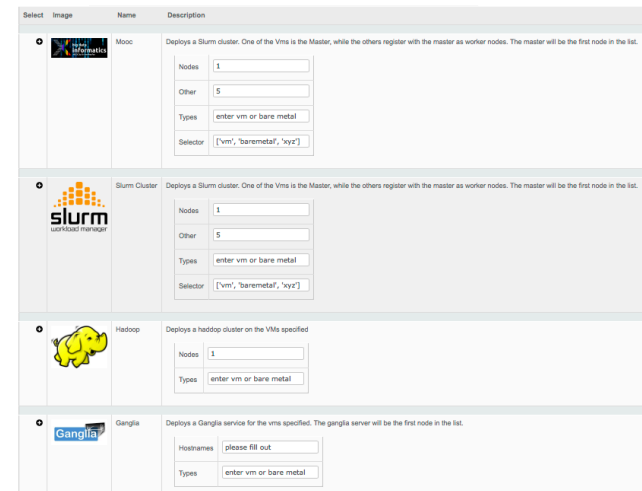


Figure 7: One click deployment of platforms and sophisticated services that could even spawn multiple resources.

tionality is developed in flask it is possible to transition it easily to other web frameworks such as web2py or django. We consider the cloudmesh API a basis for such reuse in other frameworks. Up to date information is available at cloudmesh.futuregrid.org.

6. CONCLUSION

In this paper we presented the design of a toolkit called Cloudmesh that allows to access to multiple clouds through convenient interfaces. This includes command line, a command shell, REST, as well as a graphical user interface. Cloudmesh is under active development and has shown its viability for accessing more than EC2 based clouds. Native interfaces to OpenStack, Azure, as well as any EC2 compatible cloud have been delivered and virtual machine management enabled. An important contribution of Cloudmesh is that it provides a sophisticated interface to bare metal provisioning capabilities that not only can be used by administrators, but also by authorized users. A role based authorization service makes this possible. Furthermore, we have developed a multi-cloud metrics framework that leverages information from various IaaS frameworks. Future enhancements will include network and storage provisioning.

7. ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant No. 0910812 and OCI 1025159. Some work of Cloudmesh is based on earlier work conducted as part of cyberaide.org.

8. REFERENCES

- [1] J. Diaz, G. von Laszewski, F. Wang, and G. C. Fox. Abstract Image Management and Universal Image Registration for Cloud and HPC Infrastructures. In *IEEE Cloud 2012*, Honolulu, June 2012. URL: <http://cyberaide.googlecode.com/svn/trunk/papers/12-cloud12-imagemanagement/vonLaszewski-12-IEEECloud2012.pdf>, doi:10.1109/CLOUD.2012.94.
- [2] J. Diaz, G. von Laszewski, F. Wang, A. J. Younge, and G. C. Fox. FutureGrid Image Repository: A Generic Catalog and Storage System for Heterogeneous Virtual Machine Images. In *Third IEEE International Conference on Cloud Computing Technology and Science (CloudCom2011)*, pages 560–564, Athens, Greece, 12/2011 2011. URL: <http://cyberaide.googlecode.com/svn/trunk/papers/11-cloudcom11-imagerepo/vonLaszewski-draft-11-imagerepo.pdf>, doi:10.1109/CloudCom.2011.85.
- [3] G. C. Fox, G. von Laszewski, J. Diaz, K. Keahey, J. Fortes, R. Figueiredo, S. Smallen, W. Smith, and A. Grimshaw. *Contemporary HPC Architectures*, chapter FutureGrid - a reconfigurable testbed for Cloud, HPC and Grid Computing. draft edition, 2012. URL: <http://cyberaide.googlecode.com/svn/trunk/papers/pdf/vonLaszewski-12-fg-bookchapter.pdf>.
- [4] T. R. Furlani, B. L. Schneider, M. D. Jones, J. Towns, D. L. Hart, S. M. Gallo, R. L. DeLeon, C.-D. Lu, A. Ghadersohi, R. J. Gentner, A. K. Patra, G. von Laszewski, F. Wang, J. T. Palmer, and N. Simakov. Using XDMoD to Facilitate XSEDE Operations, Planning and Analysis. In *Proceedings of the Conference on Extreme Science and Engineering Discovery Environment: Gateway to Discovery*, XSEDE '13, pages 46:1–46:8, New York, NY, USA, 2013. ACM. doi:10.1145/2484762.2484763.
- [5] Gregor. Cloudmesh on Github. Web Page. URL: <http://cloudmesh.github.io/cloudmesh/>.
- [6] K. Keahey, P. Armstrong, J. Bresnahan, D. LaBissoniere, and P. Riteau. Infrastructure outsourcing in multi-cloud environment. In *Proceedings of the 2012 Workshop on Cloud Services, Federation, and the 8th Open Cirrus Summit*, FederatedClouds '12, pages 33–38, New York, NY, USA, 2012. ACM. URL: <http://doi.acm.org/10.1145/2378975.2378984>, doi:10.1145/2378975.2378984.
- [7] T. Kurze, M. Klems, D. Bermbach, A. Lenk, S. Tai, and M. Kunze. Cloud federation. In *CLOUD COMPUTING 2011, The Second International Conference on Cloud Computing, GRIDS, and Virtualization*, pages 32–38, 2011.
- [8] Rightscale inc. [Online]. URL: <http://www.rightscale.com/>.
- [9] G. von Laszewski. Cmd3. Github Documentation and Code. URL: <http://cloudmesh.futuregrid.org/cmd3/>.
- [10] G. von Laszewski, J. Diaz, F. Wang, and G. C. Fox. Comparison of Multiple Cloud Frameworks. In *IEEE Cloud 2012*, Honolulu, HI, June 2012. URL: http://cyberaide.googlecode.com/svn/trunk/papers/12-cloud12-cloudcompare/laszewski-IEEECloud2012_id-4803.pdf, doi:10.1109/CLOUD.2012.104.
- [11] G. von Laszewski and G. Fox. *TBD*, chapter The FutureGrid Testbed for Big Data, page TBD. TBD, Indiana University, 2014.
- [12] G. von Laszewski, G. C. Fox, F. Wang, A. J. Younge, Kulshrestha, G. G. Pike, W. Smith, J. Voeckler, R. J. Figueiredo, J. Fortes, K. Keahey, and E. Deelman. Design of the FutureGrid Experiment Management Framework. In *Proceedings of Gateway Computing Environments 2010 (GCE2010) at SC10*, New Orleans, LA, Nov. 2010. IEEE. URL: <http://cyberaide.googlecode.com/svn/trunk/papers/10-FG-exp-GCE10/vonLaszewski-10-FG-exp-GCE10.pdf>, doi:10.1109/GCE.2010.5676126.
- [13] G. von Laszewski, H. Lee, J. Diaz, F. Wang, K. Tanaka, S. Karavinkoppa, G. C. Fox, and T. Furlani. Design of an Accounting and Metric-based cloud-shifting and Cloud-seeding Framework for Federatedclouds and Bare-metal Environments. In *Proceedings of the 2012 Workshop on Cloud Services, Federation, and the 8th Open Cirrus Summit*, FederatedClouds '12, pages 25–32, New York, NY, USA, 2012. ACM. doi:10.1145/2378975.2378982.