Design of an Accounting and Metric-based Cloud-shifting and Cloud-seeding framework for Federated Clouds and Bare-metal Environments

Gregor von Laszewski[1]*, Hyungro Lee[1], Javier Diaz[1], Fugang Wang[1], Koji Tanaka[1], Shubhada Karavinkoppa[1], Geoffrey C. Fox[1], Tom Furlani[2]

[1]Indiana University
2719 East 10th Street
Bloomington, IN 47408. U.S.A.
laszewski@gmail.com

[2]Center for Computational Research
University at Buffalo
701 Ellicott St
Buffalo, New York 14203

ABSTRACT

We present the design of a dynamic provisioning system that is able to manage the resources of a federated cloud environment by focusing on their utilization. With our framework, it is not only possible to allocate resources at a particular time to a specific Infrastructure as a Service framework, but also to utilize them as part of a typical HPC environment controlled by batch queuing systems. Through this interplay between virtualized and non-virtualized resources, we provide a flexible resource management framework that can be adapted based on users' demands. The need for such a framework is motivated by real user data gathered during our operation of FutureGrid (FG). We observed that the usage of the different infrastructures vary over time changing from being over-utilized to underutilize and vice versa. Therefore, the proposed framework will be beneficial for users of environments such a FutureGrid where several infrastructures are supported with limited physical resources.

Categories and Subject Descriptors

D.4.8 [Performance]: Operational Analysis, Monitors, Measurements D.4.7 [Organization and Design]: Distributed systems

General Terms

Management, Measurement, Performance, Design, Economics.

Keywords

Cloud Metric, Dynamic Provisioning, RAIN, FutureGrid, Federated Clouds, Cloud seeding, Cloud shifting.

1. INTRODUCTION

Batch, Cloud and Grid computing build the pillars of today's modern scientific compute environments. Batch computing has traditionally supported high performance computing centers to better utilize their compute resources with the goal to satisfy the many concurrent users with sophisticated batch policies utilizing a number of well managed compute resources. Grid Computing and its predecessor meta-computing elevated this goal by not only introducing the utilization of multiple queues accessible to the users, but by establishing virtual organizations that share resources among the organizational users. This includes storage and compute resources and exposes the functionality that users need as services. Recently, it has been identified that these models are too restrictive, as many researchers and groups tend to develop and deploy their own software stacks on computational resources to build the specific environment required for their experiments. Cloud computing provides here a good solution as it introduces a level of abstraction that lets the advanced scientific community assemble their own images with their own software stacks and deploy them on large numbers of computational resources in clouds. Since a number of Infrastructure as a Service (IaaS) exist, our experience [1] tells us the importance of offering a variety of them to satisfy the various user community demands. In addition, it is important to support researchers that develop such frameworks further and may need more access to the compute and storage hardware resources than is provided by the current IaaS frameworks. For this reason, it is also important to provide users with the capabilities of staging their own software stack. This feature has also been introduced by other test-beds. This includes OpenCirrus [2], EmuLab [3], Grid5000 [4] and FutureGrid [5]. Within FutureGrid we developed a sophisticated set of services that simplify the instantiation of images that can be deployed on virtualized and non-virtualized resources contrasting our efforts.

The work described here significantly enhances the services developed and described in our publications about FutureGrid focusing on dynamic provisioning supported by image management, generation, and deployment [1] [6].

In this paper, we enhance our services in the following aspects:

a) Implementation of a uniform cloud metric framework for Eucalyptus 3 and OpenStack Essex.

b) Design of a flexible framework that allows resource re-allocation between various IaaS frameworks, as well as bare-metal.

c) Design of a meta-scheduler that re-allocates resources based on metric data gathered from the usage of different frameworks.

d) Targeted prototype development and deployment for FutureGrid.

The paper is organized as follows. In Section 2, we introduce the current state of Cloud Metrics as used in various IaaS frameworks. In Section 3 we give a short overview of FutureGrid. In Section 4 we present our elementary requirements that are fulfilled in our design, presented in Section 5. In Section 6 we outline the current status of our efforts and conclude our paper in Section 7.

2. ACCOUNTING SYSTEMS

Before we can present our design it is important to review existing accounting systems, as they will become an integral part of our design and solution. This not only covers accounting systems for clouds, but also for HPC and Grid computing as motivated by user needs and the ability of FutureGrid to provide a testbed for clouds, HPC, and Grid computing as discussed in more detail in Section 3.

Accounting systems have been put into production since the early days of computing stemming back to the mainframe, which introduced batch processing, but also virtualization. The purpose of such an accounting system is manifold, but one of its main purposes is to define a policy that allows the shared usage of the resources:

- *Enable tracking of resource usage* so that an accurate picture of current and past utilization of the resources can be determined and become an input to determining a proper resource policy.
- *Enable tracking of jobs and service usage by user and group* as they typically build the common unit of measurement in addition to the wall clock time as part of a resource allocation policy.
- *Enable a metric for economic charge* so that it can be integrated into resource policies as one input for scheduling jobs within the system.
- *Enable a resource allocation policy* so that multiple users can use the shared resource. The policy allows users to get typically a quota and establishes a priority order in which users can utilize the shared resource. Typically a number of metrics are placed into a model that determines the priority and order in which users and their jobs utilize the resource.
- *Enable the automation of the resource scheduling* task to a systems service instead of being conducted by the administrator.

One of the essential ingredients for such an accounting system are the measurements and metrics that are used as input to the scheduling model and is part of the active computer science research since 1960 with the advent of the first mainframes.

2.1 High Performance Computing

As High Performance Computing (HPC) systems have always been shared resources, batch systems usually include an accounting system. Typically metrics that are part of scheduling policies include number of jobs run by a user/group at a time, overall time used by a user/group on the HPC system, wait time for jobs to get started, size of the jobs, scale of the jobs, and more. Many batch systems are today available and include popular choices such as Moab which originated from Maui, SLURM [7], Univa Grid Engine [8] which originated from CODINE [9], PBS [10], and others.. Recently many of these vendors have made access to manipulation of the scheduling policies and the resource inventory, managed by the schedulers, much easier by adding Graphical user interfaces to them [10-12]. Many of them have also added services that provide cloud-bursting capabilities by submitting jobs for example to private or public clouds such as AWS.

One of the more popular accounting systems with the community is Gold [13]. Gold introduces an economical charge model similar to that of a bank. Transactions such as deposits, charges, transfers, and refunds allow easy integration with scheduling tools. One of the strength of Gold was its free availability and the possibility to integrate it with Grid resources. Unfortunately, the current maintainers of Gold have decided to discontinue its development and instead provide an alternative as a paid solution. It has to be seen if the community will continue picking up Gold or if they switch to the new system.

An interesting projects that has recently been initiated is the XDMod project [14] that is funded by NSF XD and is integrated into the XSEDE project. One of the tasks of this project includes the development of a sophisticated framework for analyzing account and usage data within XSEDE. However, we assume this project will develop an open source version that can be adapted for other purposes. This component contains an unprecedented richness of features to view, and creates reports based on user roles and access rights. It also allows the export of the data through Web services.

2.2 Grids

Accounting systems in Grids were initially independent from each other. Each member of a virtual organization had, by design, its own allocation and accounting policies. This is verifiable by the creation of the earliest virtual organization termed GUSTO [15], but also in more recent efforts such as the TeraGrid [16], XSEDE [17] and the OpenScience Grid [18]. Efforts were put in place later to establish a common resource usage unit to allow trading between resources, as for example in TeraGrid and XSEDE. The earliest metric to establish usage of Grid services outside of such frameworks in an independent fashion was initiated by von Laszewski et al. [19] for the usage of GridFTP and later on enhanced and integrated by the Globus project for other Grid services such as job utilization. Other systems such as Nimrod [20] provided a platform to the users in the Grid community that introduced economical metrics similar to Gold and allowed for the creation of trading and auction based systems. They have been followed up by a number of research activities [21] but such systems have not been part of larger deployments in the US.

2.3 Clouds

The de facto standard for clouds has been introduced by Amazon Web Services [22]. Since the initial offering, additional IaaS frameworks have become available to enable the creation of privately managed clouds. As part of these offering, we have additional components that address accounting and usage metrics. We find particularly relevant the work conducted by Amazon [23], Eucalyptus [24], Nimbus [25], OpenStack [26], and OpenNebula [27]. Other ongoing community activities also contribute in the accounting and metric area, most notably by integrating GreenIT [28, 29]. In addition, some of these cloud platforms can be enhanced by external *monitoring* tools like Nagios [30] and Ganglia [31].

For IaaS frameworks we make the following observations.

Amazon CloudWatch [23] provides real-time monitoring of resource utilization such as CPU, disk and network. It also enables users to collect metrics about AWS resources, as well as

publish custom metrics directly to Amazon CloudWatch. *Eucalyptus* enables, since version 3.0, usage reporting as part of its resource management [24, 32]. However, it does not provide a sophisticated accounting system that allows users and administrators to observe details about particular VM instances. To enable this third party tools such as Nagios, Ganglia and log file analyzers [24] have to be used. An integrated sophisticated and convenient framework for accounting is not provided by default.

Nimbus claims to support per-client usage tracking and per-user storage quota in its image repository and in Cumulus (Nimbus storage system) as accounting features. The per-client usage tracking provides information of requested VM instances and historical usage data. The per-user storage quota enables restriction of file system usage. Nimbus also uses Torque resource manager for gathering accounting logs. For monitoring features, Nimbus utilizes Nagios and Cloud Aggregator, which is a utility to receive system resource information.

OpenNebula has a utility named OpenNebula Watch [27] as an accounting information module. It stores activities of VM instances and hosts (clusters) to show resource utilization or charging data based on the aggregated data. OpenNebula Watch requires database handler like sequel, sqlite3 or MySQL to store the accounting information. It checks the status of hosts so physical systems can be monitored, for example, CPU and memory except network.

OpenStack is currently under heavy development in regards to many of its more advanced components. An on-going effort for developing accounting systems of OpenStack exists which is named Efficient Metering or ceilometer. It aims to collect all events from OpenStack components for billing and monitoring purposes [33]. This service will measure general resource attributes such as CPU core, memory, disk and network as used by the nova components. Additional metrics might be added to provide customization. Efficient Metering is planned to be released in the next version of Openstack (Folsom) late in 2012. Besides this effort, other metric projects include several billing projects such as Dough [34] and third party OpenStack billing plugin [35].

Microsoft Azure has a software called System Center Monitoring Pack that enables the monitoring of Azure applications [36]. According to Microsoft, the monitoring pack provides features such monitoring da and performance data, with integration to Microsoft supported products such as Azure, .NET. The performance monitoring can also be enabled by using some tools like Powershell cmdlets for Windows Azure [37] and Azure Diagnostics Manager 2 from Cerebrata [38]. The monitoring data can be visualized using System Center Operation Manager Console.

Google Compute Engine is an IaaS product launched end of June, 2012 and still under development [39]. Google currently supports several options for networking and storage while managing virtual machines through the compute engine. Presently, there is no accounting APIs for Google Compute Engine, but there is a monitoring API for Google App Engine. It delivers a usage report for displaying resource utilization of instances in the administration console [40] and provides a runtime API [41] to retrieve measured data from the application instances such as CPU, memory, and status. We expect that similar functionality will become available for the Google Compute Engine as well.

3. FUTUREGRID: AS A TESTBED FOR FEDERATED CLOUD RESEARCH

FutureGrid [42] provides a set of distributed resources totaling more than 4300 compute cores. Resources include a variety of different platforms allowing users to access heterogeneous distributed computing, network, and storage resources. Services to conduct HPC, Grid, and Cloud projects including various IaaS and PaaS are offered. Interesting interoperability and scalability experiments that foster research in many areas, including federated clouds, becomes possible due to this variety of resources and services. Users can experiment with various IaaS frameworks at the same time, and also integrate Grid and HPC services that are of special interest to the scientific community. One important feature of FutureGrid is that its software services can make use of the physical resources through both virtualization technologies and dynamic provisioning on bare-metal. This feature is provided by our software called *Rain* [1, 6], which allows us to *rain* a software stack and even the OS onto a compute server/resource. Authorized users have access to this feature that is ideal for performance experiments. Via the help of *Rain,* we can now devise a design and implementation that can re-allocate compute servers into various clouds determined by user demand. We refer to this new functionality as *cloud shifting*.

4. REQUIREMENTS

In [1] we presented qualitative and quantitative evidence that users are experimenting with a variety of IaaS frameworks. To support this need, we have instantiated multiple clouds based on multiple IaaS frameworks on distributed compute clusters in FG. However, the association of compute servers to the various IaaS frameworks is currently conducted manually by the system administrators through best effort. As this can become a labor intensive process, readjustments based on utilization needs occur infrequently, or not at all as some clusters have been dedicated to a particular IaaS framework regardless of utilization. However, our operational experience shows that readjustments are desirable while observing the usage patterns of over 240 projects hosted on FutureGrid. One such use pattern arises from educational classes in the distributed computing area. We observe that classes cycle through topics to teach students about HPC, Grid, and Cloud computing. When teaching cloud computing they also introduce multiple cloud IaaS frameworks. Thus, the demand to access the resources one after another is a logical consequence based on the way such classes are taught. However, this leads to resource starvation as at times certain services offered are underutilized, while others are over utilized.

Additionally, we observe that some projects utilize the resources in a federated fashion either while focusing on federation within the same IaaS framework [43], but more interestingly to federate between IaaS frameworks while focusing on scientific workflows that utilize cycle scavenging [44] or select frameworks that are most suitable for a particular set of calculations as part of the workflow [45]. These projects do not take into account that it is possible to conduct cloud shifting instead of scavenging resulting in a simplification of the development and utilization aspect for application developers.

In a coordinated response to our observations, we derived the following requirements that shape the design of the services in support of cloud federation research:

- *Support for multiple IaaS*: This includes OpenStack, Nimbus, Eucalyptus, and OpenNebula. Furthermore, we

would like to integrate with AWS and potentially other clouds hosted outside of FG.

- *Support for bare-metal provisioning to the privately managed resources*: This will allow us to *rain* custom designed software stacks on OS on demand onto each of the servers we choose.
- *Support for dynamic adjustment of service assignments*: The services placed on a server are not fixed, but can change over time via *Rain* [1, 6].
- *Support for educational class patterns*: Compute classes often require a particular set of services that are accessed by many of its members concurrently leading to spikes in the demand for one service type.
- *Support for advance provisioning:* Sometimes users know in advance when they need a particular service motivating the need for the instantiation of services in advance. This is different from advance reservation of a service, as the service is still shared by the users after the provisioning has taken place. Such a service will help to reduce resource starvation.
- *Support for advance reservation:* Some experiments require the exclusive access to the services.
- *Support for automation:* Managing such an environment should be automatized as much as possible.
- *Support for inter-cloud federation experiments:* Ability to access multiple IaaS instances at the same time.
- *Support for diverse user communities*: Users, Administrators, Groups, and services are interested in using the framework. These groups require different access rights and use modalities.

We intend to implement this design gradually and verify it on FG. The resulting software and services will be made available in open source so others can utilize them as well.

Due to these requirements we must support four very important functions of our framework. These functions include:

Cloud-bursting enables access to additional resources in other clouds when the demand for computing capacity spikes. It outsources services in case of over-provisioning, or inter-cloud federation enabling to use compute or storage resources across various clouds.

Cloud-seeding enables the instantiation of new cloud frameworks within FutureGrid.

Cloud-shifting enables moving (or re-allocating) compute resources between the various clouds and HPC.

Resource Provisioning is a basic functionality to enable cloud-seeding and -shifting as it allows the dynamic provisioning of the OS and software stack on bare-metal.

5. DESIGN

Before we explain our architecture, we have to point out some features of the resource and service fabric that are an integral part of our design. We assume that the *Resource Fabric* consists of a resource pool that contains a number of compute services. Such services are provided either as a cluster or as part of a distributed network of workstations (NOW). The resources are grouped based on network connectivity proximity. This will allow the creation of regions within cloud IaaS environments to perform more efficiently among its servers. We assume a rich variety of services offered in the *Service Fabric*. This includes multiple IaaS, PaaS frameworks, and HPC environments. Instead of assuming that there can only be one cloud for a particular IaaS framework, we envision multiple independent clouds. This assumption potentially allows users to host their own privately managed clouds and also integrate them with public clouds. We have already deployed such an infrastructure as part of the FutureGrid, allowing users to access a variety of preconfigured clouds to conduct interoperability experiments among the same IaaS and also different IaaS frameworks, as well as the inclusion of dedicated HPC services.

Having access to such a comprehensive environment opens up a number of interesting design challenges. We observe that our operational mode is significantly enhanced in contrast to other academic clouds that typically only install a single IaaS framework on their resource [46, 47]. Thus, such environments cannot offer by themselves the comprehensive infrastructure needed to conduct many of the topics that arise in cloud federation.

One of the questions we need to answer is how we can best utilize such an environment that supports inter-cloud and bare-metal demands posed by the users as we have practically observed in FutureGrid and how we can integrate these requirements into a software architecture.

We have designed a software architecture to address the requirements presented earlier. We distinguish the user layer allowing administrators, but also users (and groups of users) to interact with the framework. In addition, we point out that Web services can interact with it to develop third party automated tools and services leveraging the capabilities. Access to the various functions is provided in a secure fashion. Due to the diverse user communities wishing to use the environment, our design supports a variety of access interfaces including command line, dashboard, web services, as well as libraries and APIs.

An important aspect is to be able to integrate existing and future information services to provide the data to guide dynamic and automatic resource provisioning, cloud–bursting, cloud-seeding, and cloud-shifting. Due to this reason, we allow in our design the integration of events posted by services such as Inca, Ganglia, and Nagios. Moreover, we obtain information from the running clouds and, when the provided information is not sufficient, we will be able to ingest our own information by analyzing log files or other information obtained when running a cloud. For clouds, we also host an instance archive that allows us to capture traces of data that can be associated with a particular virtual machine instance. A metric archive allows the registration of a self-contained service that analyses the data gathered while providing a data series according to the metric specified. Metrics can be combined and can result in new data series.

At the center of this design is a comprehensive *RAIN* service Layer. *Rain* is an acronym for *R*untime *A*daptable *IN*sertion service signifying services that on the one hand adapt to runtime conditions and on the other allow inserting or dynamically provisioning software environments and stacks. We use the terms *rain* and *raining* to refer to the process of instantiating services on the resource and service fabrics. In this analogy, we can *rain* onto a cluster services that correspond to an IaaS, a PaaS, or a HPC batch system. *Rain* can be applied to virtualized and non-virtualized machine images and software stacks. In addition, *Rain* can also be used to move resources between already instantiated environments, hence supporting cloud-shifting. The most elementary operation to enable cloud-seeding

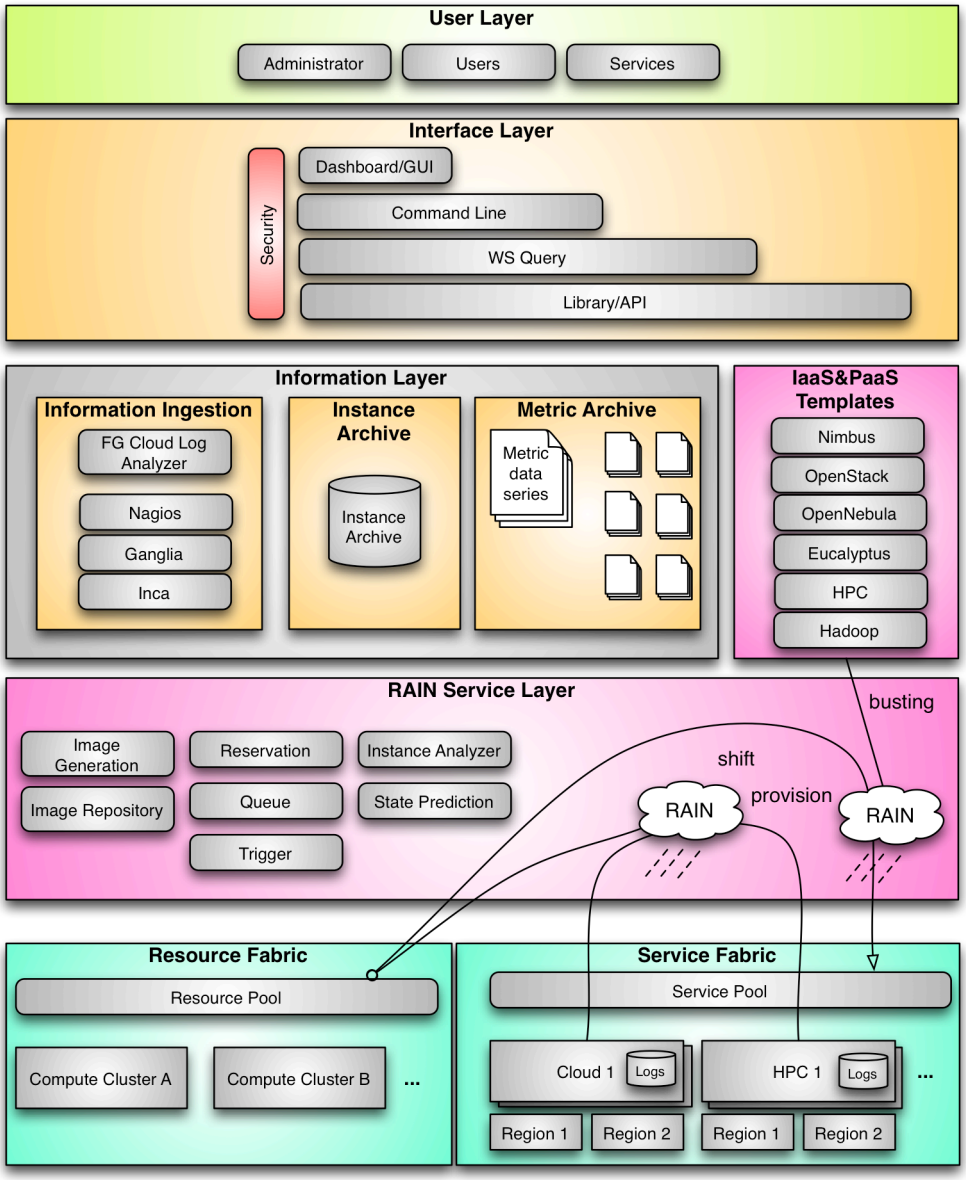

Figure 1: Design of the *rain*-based federated cloud management services.

and cloud-shifting is to provision the software and services onto the resources. We have devised this elementary operation and introduced in [6] and can now build upon it. In our past effort, we took on the problem of image management. In this work we focus on cloud-shifting, which is a logical extension in order to satisfy our users' needs.

Image Management. Rain allows us to dynamically provision images on IaaS and HPC resources. As users need quite a bit of sophistication to enable a cross platform independent image management, we have developed some tools that significantly simplify this problem. This is achieved by creating template images that are stored in a common image repository and adapted according to the environment or IaaS framework in which the image is to be deployed. Hence, users have the ability to setup experiment environments that provide similar functionality in different IaaS such as OpenStack, Eucalyptus, Nimbus, and HPC. Our image management services support the entire image lifecycle including generation, storage, reuse, and registration. Furthermore, we have started to provide extensions for image usage monitoring and quota management.

Cloud Shifting enables the re-allocation of compute resources within the various clouds and HPC. To enable cloud-shifting we have introduced a number of low-level tools and services that allow the re-allocation of resources from an IaaS or HPC service to another. A typical cloud-shifting request follows these steps:

1. Identify which resources should be moved (re-allocated) as part of the shift. This can be done by simply providing the names of the resources or by letting the service identify them according to service level agreements and specifiable requirements, such as using free nodes which have some specific characteristics.
2. De-register the resources from the service they are currently registered on. This involves identifying running jobs/VMs on the selected resource. If they exist, the service will wait a fixed amount of time for them to finish or it will terminate them. The behavior is selected when placing the request. Once a resource becomes available it will be placed into an *available resource pool*.

3. Pick resources from the available resource pool and *rain* the needed OS and other services onto each resource (if not already available).
4. Register the resources with the selected service and advertise their availability.
5. The resources are now integrated in the service and can be used by the users.

Cloud-Seeding allows us to deploy a new service such as cloud infrastructure, from scratch, in FutureGrid. Thus, cloud-seeding enhances and makes use of the previously described cloud-shifting. A cloud-seeding request includes:

1. Install the new service (cloud or HPC infrastructure) in the selected resources.
2. Set up the new service with some predefined configuration or following some given specifications. These specifications may include private IP range, ACL policies, and users' profiles.
3. Make use of cloud-shifting to add resources to the newly created service.
4. Announce and advertise the availability of the new service to the users and other services.

This is not a simple process, because it requires a great deal of planning and knowledge about the available infrastructure. Currently, we do this planning step by hand, but we intend to further automatize it as much as possible

Queue Service. We anticipate that users may have demands that cannot be immediately fulfilled by using a single request of the cloud shifting or cloud seeding services. Therefore, our design includes the introduction of a queuing service that can coordinate multiple requests. In this way, users can create a workflow that will subsequently call different services to create the desired environment. Such a queuing service could be hidden from the users and integrate with cloud-bursting services to integrate additional resources in case of overprovisioning. Once no additional resources are available requests are queued.

Reservation Service. Our design also includes the introduction of a reservation service to satisfy users with definite requests to be fulfilled at predefined times. This is the case for tutorials, classes, and regularly executed experiments.

State Prediction Service. This service will provide accounting and usage information, as well as, access to customized metrics while monitoring our different services to predict their anticipated usage. For cloud IaaS frameworks, our instance database and instance analyzer (that we developed) will collect valuable input of the resource and service fabrics.

Metrics. Elementary inputs to our prediction service are the metrics to guide our framework. These metrics are fed by elementary information in regards to job and virtual machine traces.

Traditional computing systems provide common resource metrics such as CPU, memory, storage, network bandwidth, and electricity utilization.

In case of VMs, we have to expand these metrics with VM specific information such as VM state, size, type, OS, memory, disk, CPU, kernel, Network IP, owner, and label. In addition, we are concerned with how much time it costs to create the VM, transfer it to a resource, instantiate and dynamically provision it, as well as bringing it in a state that allows access by the user. Furthermore, once the machine is shut down, we need to account

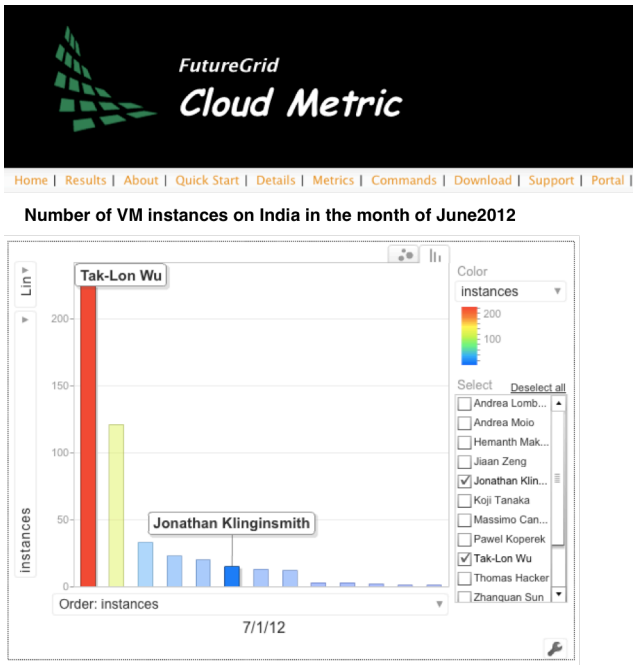

Figure 2: Screenshot of our Cloud Instance Analyzing

for the shutdown time and eventual cleanup or removal of the VM. Naturally we also need to keep track of which user, group or project instantiated the VM and if the image is a replication run in parallel on other resources in the fabric.

When dealing with services that are dependent on performance metrics, we also have to deal with periodicity of the events and filter out events not only based potentially on a yearly, monthly, weekly, daily, hourly, minute or per second basis, but to eliminate events that do not contribute significantly to the trace of a virtual machine image. We have practically devised such a service for Eucalyptus that reduced four million log events to about 10000 trace events for virtual machine images. This allows us to query needed information for our predictive services in milliseconds rather than hours of reanalyzing such log entries over and over again. Hence, our design is not only to retrieve standard information such as average, sum, minimum and maximum, as well as count of VM related events, but it can also input this data efficiently into a time series analysis and predictive service. In addition, we have integrated metrics for OpenStack and are in the process of expanding to Nimbus. Clearly, this framework is a sophisticated tool in support of federated heterogeneous and homogeneous clouds.

6. STATUS AND IMPLEMENTATION

As already pointed out, we have developed the basic infrastructure to support *rain* by enabling the image management services. These services are in detail documented in [1, 6, 42]. Recently we started the development of *rain* services that address the issue of cloud-shifting. We developed the ability to add and remove resources dynamically to and from Eucalyptus, OpenStack and HPC services. This allows us to easily move resources between OpenStack, Eucalyptus, and HPC services. We used this service to shift resources in support of a summer school held at the end of July 2012 where more than 100 participants were taught a variety of IaaS and PaaS frameworks. Within a short period of time we were able to adapt

our resource assignments to more closely serve the requirements of the projects executed at the time. As currently, some features in Nimbus are missing that are necessary to integrate with our framework, FutureGrid is also funding the Nimbus project to enhance their services so they will allow similar features as other IaaS frameworks already provide in order to support our image management framework. In parallel, we have significantly contributed towards the analysis of instance data for Eucalyptus and OpenStack clouds. Such data is instrumental for our planned predictive services. This effort includes the creation of a comprehensive database for instance traces that records important changes conducted as part of the VM instance runtime documented in our design section. A previous analysis effort that analyses log files in a repeated fashion was designed and implemented by von Laszewski, Wang, and Lee, replacing an effort that allows the ingestion and extraction of important log files from newly created log events [48]. As a result, we were able to significantly reduce the log entries, which led to a speedup of our analyzing capabilities from hours to milliseconds. In addition, we made the framework independent from web frameworks and chart display tools. A convenient command shell that can also be accessed as a command line tool was added to allow for interactive sampling and preparation of data. Web services, as well as a simple graphical interface to this data will be available (see Figure 2). At the same time the code was significantly reduced and modularized so that future maintenance and enhancements become easier. Examples for data currently presented in our Web interface are based on the utilization of several metrics. This includes total running hours of VM instances; total number of VM instances in a particular state and time interval; CPU core, memory and disk allocations; delay of launching and termination requests, the provisioning Interval, and geographical locations of VM instances. Metrics projecting a per user, per group, or per project view, metrics per cloud view, as well as metrics for the overall infrastructure and metrics related to the resource and service fabric are under development. Additional metrics such as traffic intensity for a particular time period [49, 50] are also useful in considering optimized utilization. Future activities will also include our strategy to use DevOps frameworks that we started from the beginning of the project and have also been independently been used by the FutureGrid community [51]. Clearly our framework can also be beneficial for integrative cloud environments such as CometCloud [52].

7. CONCLUSION

In this paper, we have presented a design of a federated cloud environment that is not focused singly on supporting just an IaaS framework. Our understanding of federation includes various IaaS frameworks on potentially heterogeneous compute resources. In addition, we are expanding our federated cloud environment to include and integrate traditional HPC services. This work is a significant enhancement to our earlier work on dynamic image generation and provisioning in clouds and bare-metal environments by addressing challenges arising in cloud seeding and cloud shifting. One of the other contributions of this paper is the creation of an accounting and metric framework that allows us to manage traces of virtual machine instances. This framework will be an essential component towards automating cloud-shifting and seeding as projected by our architectural design. We welcome additional collaborators to contribute to our efforts and to use FutureGrid.

8. ACKNOWLEDGEMENTS

This material is based upon work supported in part by the National Science Foundation under Grant No. 0910812 and 1025159. We like to thank the members of FG for their help and support. We like to thank the Eucalyptus team for their willingness and excellent help in letting us use their commercial product.

9. REFERENCES

[1] von Laszewski, G., Diaz, J., Wang, F. and Fox, G. C. Comparison of Multiple Cloud Frameworks. In *Proceedings of the IEEE CLOUD 2012, 5th International Conference on Cloud Computing* (Honolulu, HI, USA, 24-29 June, 2012). Doi 10.1109/CLOUD.2012.104.

[2] Avetisyan, A. I., Campbell, R., Gupta, I., Heath, M. T., Ko, S. Y., Ganger, G. R., Kozuch, M. A., O'Hallaron, D., Kunze, M., Kwan, T. T., Lai, K., Lyons, M., Milojicic, D. S., Hing Yan, L., Yeng Chai, S., Ng Kwang, M., Luke, J. Y. and Han, N. Open Cirrus: A Global Cloud Computing Testbed. *Computer*, 43, 4 2010), 35-43. Doi 10.1109/mc.2010.111.

[3] White, B., Lepreau, J., Stoller, L., Ricci, R., Guruprasad, S., Newbold, M., Hibler, M., Barb, C. and Joglekar, A. An Integrated Experimental Environment for Distributed Systems and Networks. In *Proceedings of the Proceedings of the 5th Symposium on Operating Systems Design & Implementation* (December 2002, 2002).

[4] Grid5000 Home Page, http://www.grid5000.fr/.

[5] G. von Laszewski, G. C. Fox, Fugang Wang, A. J. Younge, A. Kulshrestha, G. G. Pike, W. Smith, J. Vöckler, R. J. Figueiredo, J. Fortes and K. Keahey. Design of the FutureGrid experiment management framework. In *Proceedings of the Gateway Computing Environments Workshop (GCE) at SC10* (New Orleans, LA, 14-14 Nov. 2010, 2010). IEEE. Doi 10.1109/GCE.2010.5676126

[6] Diaz, J., von Laszewski, G., Wang, F. and Fox, G. Abstract Image Management and Universal Image Registration for Cloud and HPC Infrastructures. In *Proceedings of the IEEE CLOUD 2012, 5th International Conference on Cloud Computing* (Honolulu, HI, 2012). IEEE. Doi 10.1109/cloud.2012.94.

[7] Yoo, A., Jette, M. and Grondona, M. SLURM: Simple Linux Utility for Resource Management. in Job Scheduling Strategies for Parallel Processing, Lecture Notes in Computer Science, vol 2862, p. 44-60, Springer Berlin / Heidelberg, 2003.

[8] Univa Grid Engine, http://www.univa.com/products/grid-engine/.

[9] Genias CODINE: Computing in distributed networked environments (1995), http://www.genias.de/genias/english/codine.html.

[10] Altair PBS, http://www.pbsworks.com/.

[11] Moab, http://www.adaptivecomputing.com/products/hpc-products/.

[12] Bright-Computing Cluster Manager, http://www.brightcomputing.com/Bright-Cluster-Manager.php.

[13] Adaptive-Computing Gold Allocation Manager User Guide, http://www.adaptivecomputing.com/resources/docs/gold/.

[14] XDMoD XDMoD (XSEDE Metrics on Demand), https://xdmod.ccr.buffalo.edu/.

[15] Globus-Project Globus Ubiquitous Supercomputing Testbed Organization (GUSTO), http://www.startap.net/PUBLICATIONS/news-globus2.html.

[16] Hart, D. Measuring TeraGrid: Workload Characterization for an HPC Federation. *International Journal of High Performance Computing Applications* 4(Nov. 2011), 451-465. Doi 10.1177/1094342010394382.

[17] XSEDE: Extreme Science and Engineering Discovery Environment, https://http://www.xsede.org.

[18] OSG Open Science Grid, http://www.opensciencegrid.org.

[19] von Laszewski, G., DiCarlo, J. and Allcock, B. A Portal for Visualizing Grid Usage. *Concurrency and Computation: Practice and Experience*, 19, 12 (presented in GCE 2005 at SC'2005 2007), 1683-1692. Doi

[20] Abramson, D., Foster, I., Giddy, J., Lewis, A., Sosic, R., Sutherst, R. and White, N. The Nimrod Computational Workbench: A Case Study in Desktop Metacomputing. In *Proceedings of the Proceedings of the 20th Australasian Computer Science Conference* (1997).

[21] Buyya, R., Yeo, C. S. and Venugopal, S. *Market-oriented cloud computing: Vision, hype, and reality for delivering IT services as computing utilities, in*. 2008.

[22] Amazon Amazon Web Services, http://aws.amazon.com/.

[23] Amazon Web Services Cloud Watch, http://docs.amazonwebservices.com/AmazonCloudWatch/latest/DeveloperGuide/CloudWatch_Introduction.html.

[24] Eucalyptus Eucalyptus Monitoring, http://open.eucalyptus.com/wiki/EucalyptusMonitoring_v1.6.

[25] Nimbus-Project Per Client Tracking, http://www.nimbusproject.org/docs/current/features.html.

[26] OpenStack Multi-Tenant Accounting, http://wiki.openstack.org/openstack-accounting?action=AttachFile&do=get&target=accounts.pdf.

[27] OpenNebula OpenNebula Watch - Accounting and Statistics 3.0, http://opennebula.org/documentation:archives:rel3.0:acctd_conf.

[28] Beloglazov, A., Buyya, R., Lee, Y. C. and Zomaya, A. A Taxonomy and Survey of Energy-Efficient Data Centers and Cloud Computing Systems2010). Doi

[29] Berl, A., Gelenbe, E., Di Girolamo, M., Giuliani, G., De Meer, H., Dang, M. Q. and Pentikousis, K. Energy-Efficient Cloud Computing2010). Doi

[30] Barth, W. *Nagios: System and Network Monitoring*. No Starch Press, San Francisco, CA, USA, 2006.

[31] Massie, M. L., Chun, B. N. and Culler, D. E. The ganglia distributed monitoring system: design, implementation, and experience. *Parallel Computing*, 30, 7 2004), 817 - 840. Doi 10.1016/j.parco.2004.04.001.

[32] Eucalyptus Eucalyptus 3.0.2 Administration guide, http://www.eucalyptus.com/docs/3.0/ag.pdf.

[33] OpenStack Blue print of Efficient Metering, http://wiki.openstack.org/EfficientMetering.

[34] Luo, Z. Dough, https://github.com/lzyeval/dough.

[35] OpenStack Billing Plugin for OpenStack, https://github.com/trystack/dash_billing.

[36] Microsoft Introduction to the Monitoring Pack for Windows Azure Applications

[37] Microsoft Windows Azure PowerShell Cmdlets, http://wappowershell.codeplex.com/.

[38] Red-Gate-Software Cerebrata, http://www.cerebrata.com/.

[39] Google Google Compute Engine, http://en.wikipedia.org/wiki/Google_Compute_Engine.

[40] Google Monitoring Resource Usage of Google App Engine, https://developers.google.com/appengine/docs/python/backends/overview#Monitoring_Resource_Usage.

[41] Google Runtime API for Google App Engine, https://developers.google.com/appengine/docs/python/backends/runtimeapi.

[42] von Laszewski, G., Fox, G. C., Wang, F., Younge, A. J., Kulshrestha, A., Pike, G. G., Smith, W., Voeckler, J., Figueiredo, R. J., Fortes, J., Keahey, K. and Deelman, E. Design of the FutureGrid experiment management framework. In *Proceedings of the Gateway Computing Environments Workshop (GCE), 2010 in conjunction with SC10* (New Orleans, LA, 14-14 Nov. 2010, 2010). IEEE. Doi 10.1109/GCE.2010.5676126.

[43] Keahey, K., Tsugawa, M., Matsunaga, A. and Fortes, J. Sky Computing. *Internet Computing, IEEE*, 132009), 43-51. Doi 10.1109/MIC.2009.94.

[44] Deelman, E., Singh, G., Su, M.-H., Blythe, J., Gil, Y., Kesselman, C., Mehta, G., Vahi, K., Berriman, G. B., Good, J., Laity, A., Jacob, J. C. and Katz, D. S. Pegasus: A framework for mapping complex scientific workflows onto distributed systems. *Scientific Programming Journal*, 13, 3 2005), 219-237. Doi 10.1.1.117.132.

[45] Thilina Gunarathne, Judy Qiu and Geoffrey Fox. Iterative MapReduce for Azure Cloud In *Proceedings of the CCA11 Cloud Computing and Its Applications* (Chicago, IL, April 12-13, 2011).

[46] Cornell Cornell University Red Cloud, http://www.cac.cornell.edu/redcloud/.

[47] Clemson Clemson University One Cloud, https://sites.google.com/site/cuonecloud/.

[48] Laszewski, G. v., Lee, H. and Wang, F. Eucalyptus Metric Framework (Source Code), https://github.com/futuregrid/futuregrid-cloud-metrics.

[49] Calheiros, R. N., Ranjan, R. and Buyya, R. Virtual Machine Provisioning Based on Analytical Performance and QoS in Cloud Computing Environments. In *Proceedings of the International Conference on Parallel Processing* (Washngton, DC, 2011). IEEE Computer Society. Doi 10.1109/ICPP.2011.17.

[50] Maguluri, S. T., Srikant, R. and Ying, L. *Stochastic Models of Load Balancing and Scheduling in Cloud Computing Clusters*. 2012.

[51] Klinginsmith, J., Mahoui, M. and Wu, Y. M. Towards Reproducible eScience in the Cloud. In *Proceedings of the Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on* (Nov. 29 2011-Dec. 1 2011, 2011). Doi 10.1109/CloudCom.2011.89.

[52] Kim, H. and Parashar, M. CometCloud: An Autonomic Cloud Engine. in Cloud Computing, vol p. 275-297, John Wiley & Sons, Inc., 2011.