Cloud Metrics for Academic Resource Providers

CONTENTS

Notes

Cloud Metrics for Academic Resource Providers

Gregor von Laszewski [1] [*], Hyungro Lee[1], Fugang Wang[1], Geoffrey C. Fox[1],
Thomas Furlani[2], Robert DeLeon[2], Qiao Chunming[2]
[1]Indiana University, 2719 10th Street, Bloomington, Indiana, U.S.A.
[2]SUNY at Buffalo, 701 Ellicott Street, Buffalo, New York, 14203, U.S.A.
[*]laszewski@gmail.com

ABSTRACT

Cloud computing has established itself as one of the key components in modern datacenters. This includes not only government, and for profit organizations but also academic institutions. One of the important aspects to a successfully operate and utilize clouds in any organization is the availability of sophisticated metric information and frameworks to assure a number of key operational factors are met that are measured by such metrics. In this paper we provide an overview of the different aspects that are shaping the need for cloud metrics. We especially identify the needs for cloud metrics and monitoring solutions for various user communities that are an important factor for academic cloud providers. The motivation for academic cloud metrics are shaped by application users, administrators, academic and funders. Metrics of interest support usage monitoring, failure and auditing data to identify user and system behavior. The information needs to be gathered as a mashup and exposed conveniently to the users as API, command tools, and a portal with charting capability, as well as periodically generated customizable reports. Many of the metrics presented, have been successfully used as part of the FutureGrid and FutureSystems clouds. This includes also the unified metrics service framework allowing integration of metrics from heterogeneous clouds such as OpenStack, Eucalyptus and Nimbus. Additionally, we see the importance of metrics also to be used as part of higher level platform as a services that utilize IaaS or are offered as PaaS and SaaS. Thus it is important to identify common features of existing and future services that benefit from metrics and their exposure to its users.

CCS Concepts

•Networks → Cloud computing; •General and reference → Metrics;

[*]Corresponding Author.

Keywords

Cloud Metrics, Resource Monitoring, Cloud Accounting, Metering, FutureGrid, FutureSystems, XSEDE

1. INTRODUCTION

1.1 Cloud Computing

Cloud computing has established itself as one of the key components in modern datacenters. This includes academic, government, and for profit institutions and organizations. According to the NIST definition "Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction" [**?**]. Essential characteristics include on-demand self-service, broad network access, rapid elasticity, measured service. Important service models are Software as a Service (SaaS), Software as a Service (SaaS), and Infrastructure as a Service (IaaS). Deployments include typically private clouds, community clouds, public clouds, and hybrid clouds.

1.2 Academic Cloud Computing

Academic cloud computing offers clouds to the academic community. In contrast to commercial offerings they may have some very specific properties that need to be considered. First we distinguish two different target communities. On the one hand it includes the operational IT organization that may provide e-mail, calendars, and document sharing environments to the staff members of the university. These are production oriented services that may be offered by private or public clouds. The demands for them are typically not different from any other organization running operational production services. On the other institutions are in the need to provide infrastructures, services, and platforms to the academic *research* and *education* community (see Figure 1). We will focus in this paper on the latter communities.

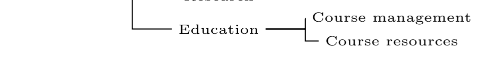


Figure 1: Academic Cloud Computing

Within the offerings to the academic community we observe the usage of the following clouds (a) commercial public clouds, Academic public clouds, and Academic private clouds. Furthermore, traditional high performance computing)HPC) is offered as a platform in many organizations and allows to support the paradigm of *HPC in the cloud* as showcased for example by the Extreme Science and Engineering Discovery Environment (XSEDE) project (see Section 2.8 for more details). The validity of HPC as a service in the cloud is reflected not only by academic offerings, but also by commercial offerings commercial offerings as demonstrated by Amazon High Performance Computing [?], IBM Platform Computing [?], and with additional examples provided in [?].

1.3 Cloud Metrics

One of the important aspects to a successfully operational cloud is the availability of sophisticated metric information and frameworks to assure a number of key operational factors. As cloud computing is typically part of a large scale data center it touches and also must integrate common data center metrics. Furthermore, it is essential to cloud specific application metrics while benchmarking a cloud at regular intervals to detect potential issues. Cloud metrics are important to a number of users interested in the various aspects dealing with a operation, use, and management of a cloud. Hence, the motivation for cloud metrics are shaped by application users, administrators, service providers and funders. Metrics include usage, failure and auditing data to identify user and system behavior. Many of the metrics presented here have been used as part of the FutureGrid and FutureSystems clouds. These general aspects are not only common to a particular IaaS framework but are common across heterogeneous cloud IaaS such as OpenStack, Eucalyptus and Nimbus including unified metrics across such deployments. Additionally, we see the importance of metrics also in higher level platform as a services delivered as part of high level cloud services designed to support various user communities and virtual organizations. Thus it is important to identify common features of existing and future services that benefit from metrics and their exposure to its users. Based on our short introduction it is clear that a more formal approach is needed to identify

1. Which metrics are useful for whom and why?
2. Which metric efforts exists?
3. Which community is targeted?
4. Which services are targeted?
5. Which resources are monitored?
6. What privacy aspects exist?

We observe that in may cases there is no distinction between academic, or non academic resource providers. Areas where there exists distinction are metrics based on charging models, and the utilization of hybrid clouds. Furthermore, we find that higher level metrics and metrics integrating into other frameworks provided as PaaS and HPC frameworks. The reason for this is that some projects may require a combination of technologies that motivate project wide metrics to not only display usage and utilization of the various frameworks and services for these projects. The pooling of resources in a heterogeneous environment provides a *computational mesh* going potentially beyond just the utilization of cloud, HPC, and Grid computing all of which may need to be integrated into a comprehensive metrics framework. Due to the limited resources in academic organizations we also often do not have (when private clouds) are utilized an unlimited resource pool. For example the XSEDE resources receive resource requests that are four times higher than the available resources. Hence, it is important to identify performance metrics and benchmarks that lead to an improvement of the applications using such resources. For academic cloud resources availability and scalability also need to be ensured to manage rapid changes (which we see for example in educational usage patterns as part of classes) in the demands for cloud resources. These characteristics of cloud computing can be satisfied with the support of performance analysis and monitoring. There are some considerations when performance analysis and monitoring are applied to cloud computing: (a) performance isolation against interference (b) availability and Scalability (c) cost effectiveness to fully utilize available resources.

As other clouds academic clouds rely on sharing of the offered services. However, just as in public clouds many users may not understand the implication and cost that is prevalent when using a particular service. Hence detailed metrics informing the user not only at time of an experiment, but access to information that helps planing such experiments are needed. Thus metrics will address and provide an essential input to several concrete challenges for academic clouds including:

Understanding - how reliable the system is and to prevent system failures. It also gives an opportunity to manage the system efficiently by knowing the performance of the system.

Getting informed - of the current status and the impact of previous activities allowing the various user communities to act appropriately and invoke defensive measures as well as being an input to btter understanding the system over time.

Estimating future requests - it discovers usage patterns and trends of system resources which allows to projection the increasing of system capacity and performance.

Reporting - Measured statistics can be viewed in different ways with various visualization tools. Several graphical tools and charts APIs can help identify which resources are consumed the most by whom, what, where, why and when.

These and other aspects are addressed in the following paper. Our paper is structured as follows. In Section 2 we introduce some major factors that need to be considered when providing a cloud monitoring and metric framework. We introduce some concrete metrics in Section 3. A brief summary of Monitoring tools and services is provided in Section 4. Section 5 showcases our hybrid cloud monitoring and metrics services that have been used in FutureGrid and are currently used in FutureSystems. Related activities are briefly summarized in Section 6. We conclude our paper while summarizing our findings in Section 7.

2. CLOUD METRIC FACTORS

When we look at a particular cloud metric, it may be shaped by a number of factors that determines the applicability and usability of the metric in general. To provide an overview of a selected number of factors fro cloud metrics we have identified the following factors of significant importance: (a) stakeholders and roles to address, (b) the frameworks to address, (c) the purpose the metrics are designed for, (d) concrete resources are targeted by the metric, (e) which sources are used for the metric, and (f) which security aspects are considered. We are summarizing these influential aspects in Figure 2 and explaining them in more detail.

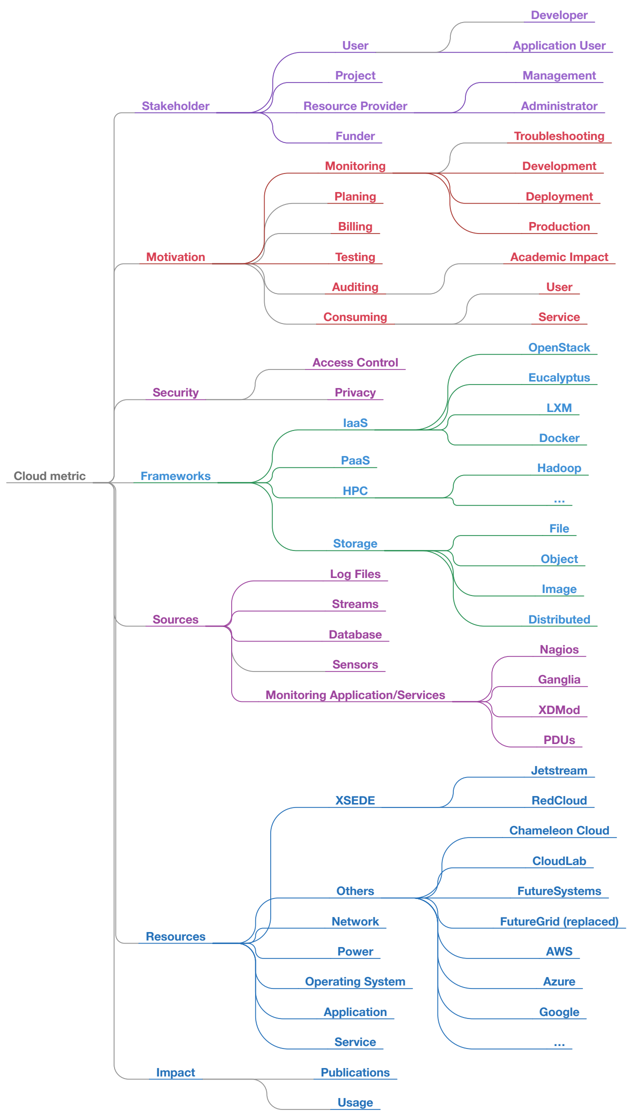


Figure 2: Overview of cloud metric factors

2.1 Stakeholder

A cloud will have a wide variety of stakeholders with their potentially specific needs in metrics. These stakeholders include (a) the users which could be application developers and application users, (b) a project in which multiple users contribute (c) the resource provider including administrators and management including the director. Furthermore, we identify the funder of the Cloud as an important stakeholder as he may have needs for specific metrics that are typically not relevant to users or administrators of the cloud but have profound impact on the goal of funding the resource. To give an example of the wide variety of metrics needed we like to make use aware that although management is interested in operational 24/7 operation of the system and statistics associated with it, concrete metrics and monitoring infrastructure as needed by the administrators to detect operational failures are typically not part of the needed metrics for a center director. On the other hand metrics to identify ethnicity and classification into scientific disciplines are of great value for center directors and Funders, while they play no importance to the operational metrics to keep the system running.

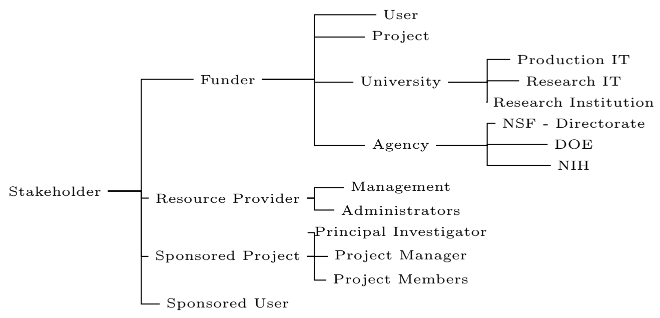


Figure 3: Cloud metric stakeholders

2.2 Access

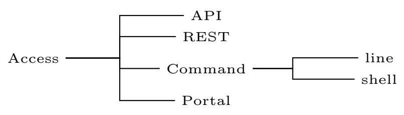


Figure 4: Cloud metric Access

2.3 Security

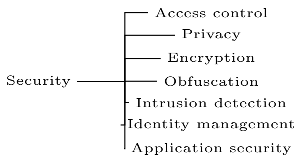


Figure 5: Cloud metric Access

2.4 Type

2.5 Motivation

Another important factor for a cloud metric is the motivation and purpose for it. We distinguish the following factors:

Monitoring. Monitoring is an important aspect for administrators but also for users. The purpose of metrics associated to monitoring allows us to observe the current system and make decisions based upon it. Naturally, this could also be services that act in behalf of the user or administrator. A center director may be

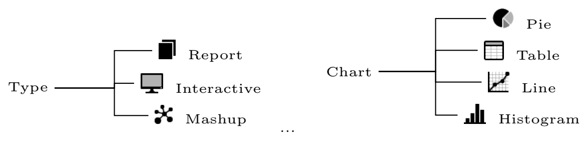


Figure 6: Cloud metric presentation types

interested in the high level aspects of monitoring as to be informed about specific catastrophic service events, while an administrator is interested in more detailed monitoring aspects that alert even on conditions that could lead to issues. Examples for monitoring includes measuring resource utilization in real time or over a period of time. Monitoring is potentially important on all layers of the Cloud infrastructure from bare metal, to IaaS, PaaS, and SaaS.

Planning. Planing is needed to assist in setting goals and developing strategies several metrics can be beneficial. This includes metrics about utilization of the resources, satisfaction by the users and other more general aspects. As academic resources are often funded by government organizations such as NSF, specific metrics that answer to broader impacts need to be addressed. An essential ingredient of planing is the ability to include performance monitoring on the cloud while exposing sophisticated metrics. This will enable the planing of efficient resource utilization from a small instance to a large virtual cluster on the cloud, performance management and monitoring are necessary for performance analysis.

Billing. As academic clouds are provided free to the academic user community upon peer reviewed projects, it is also important to avoid situations that either overstress or adversely lead to underutilization. For example in absence of billing against real monetary values, we observed that many users ignore the cost that is involved by running an actual VM. Furthermore, it is important to communicate to the users the potential estimated cost to them in order to educate them towards deleting or suspending unused resources. However, at the same time it must be assured that large enough experiments can be conducted to further some of the more challenging scientific problems.

Testing. As cloud environments and they usage in applications may be complex (at times more complex that their HPC counter parts) it is important that metrics be provided that support the testing of the functionality, the performance, and the scalability. Ideally these metrics should be available as part of automatic testing services that however may be customized for a particular application.

Consuming. Many metrics will assist the users and services to consume the cloud services efficiently. They will provide information in optimizing resource use and distribution amongst them. They will also alert towards limitations of the underlaying systems and provide insight on where application limitations my hinder adoption. A specific case is the creation of user motivated benchmarks that through their metrics can inform users about which cloud setup is particularly beneficial.

Auditing. Metrics for auditing support the process to evaluate the clouds design and effectiveness. This includes security, deployment processes, development processes and governance and oversight processes. It includes finding a proof and a trait of actions a user made while resources are being used. Observing a user's, or service's behavior should be performed by logging events and detailed information is necessary to track back any issues on a system. An important aspect or a comprehensive auditing process is that that it is conducted by an independent and unbiased observer. Auditing should not be confused with monitoring the system in real time.

2.6 Security

Due to targiteng different user communities it is obvious that smoe metrics may not be exosed to all its users and are potentially unde access control. Furthermore, users that consume cloud resources may have the desire that their information is private and may not be shared with other users. However, although it is possible to provide acess and privacy controls we have seen in government agency sponsord reserach environments a general consensus of transparency mandated by the funding agency such as NSF. A good example here is XSEDE that provides a great deal of information to its users. When looking at a metic framework for clouds we need to consider the posibility to restrict eccess through access control policies that regulate privacy concerns. Such policies need to include role based restrictions of stakeholders and metrics to be accessed by them.

2.7 Frameworks

As already pointed out in the introduction, it is important to not isolate the cloud metrics. Although they can be developed independently in many cases we need a larger picture for projects that require a heterogeneous set of services and frameworks for their scientific mission to succeed. This includes the integration of high performance computing resources and services (HPC), storage resources and services, infrastructure as a service based resources build from either OpenStack, VM Ware, or Nimbus, as well as various platform as a service environments that enable academic users to focus on the platform regardless of which underlaying resource framework is chosen. Furthermore it may be important to consider extending the cloud metrics framework to public clouds such as using AWS as this may be an integral part of the strategy to offer services to academic users and groups. Restrictions posed by some of these frameworks such as the lack of groups in the Nimbus framework make it problematic to meet the demands fro example of an academic (e.g. XSEDE) based metric framework. Thus Nimbus must provide group support in order to become meaningful in academic project based deployments while providing comprehensive metrics.

2.8 Resources

Metrics that are implicitly provided by various cloud providers are important for gaining a complete picture of a particular project. This may include the following

Commercial public clouds. These are clouds that are offered commercially or for free to academic users. While an individual se of time limited user based usage may not be of importance it becomes essential if actual money is paid to provide this service to academic users. Important may not only the amount of money paid, but to justify the academic and scientific impact [?, ?] that is derived by such efforts.

Academic public clouds. Recently NSF has provided a significant amount of funding to cloud offerings for the academic community. In order to justify its funding and to contrast it to commercial public cloud offerings metrics and evaluation criteria need to be developed that justify their existence.

Academic private clouds. Many universities have started to offer private clouds to their user community, This includes not only the use of academic efforts, but also of production clouds to support the internal IT infrastructure as part of a universities management IT infrastructure. While in the IT department operational cost and privacy are potentially dominating the metrics, in academic clouds the se in projects and their outcome metrics need to be integrated. This reflects similar metrics that we find while using commercial and academic public clouds.

Examples for publicly funded clouds include XSEDE while offering HPC services to its users, Chameleon Cloud, CloudLab, as IaaS based clouds and network experiment infrastructure, and Jetstream as Iaas and PaaS supporting cloud. Additionally the following Resources offered to academic users will have a potential impact on the definition of Metrics:

MRIs. According to the NSF web pages, the 'Major Research Instrumentation Program (MRI) [?] catalyzes new knowledge and discoveries by empowering the Nation's scientists and engineers with state-of-the-art research instrumentation. The MRI Program enables research-intensive learning environments that promote the development of a diverse workforce and next generation instrumentation, as well as facilitates academic and private sector partnerships." Some of the MRI funding supports significant computational resources offered to a particular user community. In such cases it would be beneficial if the metrics and services introduced here can be reused by such efforts and adapted accordingly.

Networks. With the advent of 100GBEthernet research environments as provided in Chameleon cloud and XSEDE it is essential that network related information be integrated into an overarching metric framework. This includes not only details of the overall use of the network, but the specific information which projects, or even which applications are using it. Access control to this information may has to be investigated in order not to expose exploitable information to the community.

Storage. Obviously many cloud applications in the bid data area require large amount of storage either as files, databases, or object stores. Metrics must be available to the users and the providers in order to assess needs and availability. Storage has traditionally been an issue in academic environments.

Power. A significant cost in operating an academic cloud is power consumption. It is important that the IT departments of the academic cloud providers put efforts in place that power usage is transparently exposed so that associations between power and services offered can be achieved. This goes beyond the measurement of poer consumption within servers, but muct inclde a more generalized approach while integrating PDU information and other available metrics that may already be available. However, typically such information is often isolated and potentially not available to the academic user community without a significant effort. Future designs of academic datacenter should keep in mind that such information ought to be transparently provided to the researchers or retrofitted accordingly.

Operating Systems. Traditionally a large amount of metric information is available as part of the operating system either in virtualized or in bare metal mode. The included services can present a great deal of information to users and to the provider. Tools such as Nagios and Ganglia provide the ability to collect and integrate the information sources.

Applications. Many academic users will develop a number of applications on available resources. Metrics that compare efficiency and effective use of such applications will be important as to evaluate prudent use of the resources. For example in some cases it may be essential to measure the performance impact of using cloud vs HPC resources. In other cases it may be more important to focus on manpower consumed to develop applications on more complex environments. Supporting performance APIs and libraries may aide in the development of metrics and their associated services such as the use of PAPI [?, ?, ?]

Services. In addition to applications we also are in the need of metrics provided by services offered to the community.

2.9 Sources

A comprehensive metrics framework consists of a data mashup of various sources of information. It will not be sufficient to just present a heterogeneous set of metrics to the users, but it will be desirable to integrate several into an easy to use framework. This has been effectively demonstrated by XDMod for HPC [?, ?, ?]. This was possible due to a sustained effort within the project to provide such integrated metrics. In the cloud IaaS area the FutureGrid project has pioneered such an integrated framework that combined multiple sources into an easy to use report generation framework [?]. It was operational for a number of years and is still actively used by the FutureSystems project. At this time it is unknown which strategies are pursued by projects such as chameleon cloud and other NSF sponsored projects. In fact there may be an advantage that such activities be continued by an external party such as FutureSystems and be adopted by other in order to be true to the independence of the auditing principle which need to be employed to evaluate the effectiveness of such infrastructures.

2.10 Scientific Impact

Many science and engineering innovations and discoveries are increasingly dependent on access to high performance computing resources [?, ?, ?]. For many researchers, this demand is met by clouds and other large-scale compute resources that cannot typically be supported by any single research group. Accordingly, dedicated large-scale computing facilities play an important role in scientific research, in which resources are shared among groups of researchers, while the facilities themselves are managed by dedicated staff. Thus, justification for their use is warranted and questions regarding the scientific impact of these resources naturally arise. Hence, it is important to not only introduce metrics measuring the actual usage, but also the impact based on for example the scientific publications that are produced by such resources. We have developed a sophisticated metric framework that delivers such impact metrics for XSEDE, NCAR, and Blue Waters [**?**, **?**, **?**]. It is most natural that this be extended to any cloud resource. However, the framework is general enough that it could apply to any resource framework and interested resource providers can contact the corresponding athor for more information. Table 1 lists the general metrics related to the impact of scientific works.

While evaluating a research computing facility, in this context, a cloud resource provider, based on how the accounting and auditing framework had been setup, as well as the mechanism of gathering publications as output of using the cloud, different metrics could be derived based on the general ones listed above in table 1. E.g., if we have user and group/project units in the accounting system, and the publications are gathered and vetted also with user/group/project information, we could derive the same metrics for each user, group/project, and compare their performance. These metrics could be calculated for a cloud resource provider as a whole entity, and the same metrics could be compared among different cloud resource provider. Based on other related data, e.g. the resource had been consumed, we could derive and compare the same metrics but in a normalized fashion - Publication count and/or citation count per M(K) CpuHours as an example.

3. CLOUD METRICS

In this section we will introduce several important examples for cloud metrics that will be important for academic cloud providers. We also will introduce several parameters that determine the sample size and population over which such metrics are invoked. The metrics that we define here are in part derived from our experience within FutureGrid and FutureSystems.

We will introduce a number of definitions that we will use throughout this section.

Period. Many metrics need to be applied on a flexible period, thus it is important to be able to define the start and end time of a metric to be applied, as well as its periodicity throughout the interval.

Realtime. Some metrics need to be applied at realtime in order to obtain immideate feedback about the system status

Time to Live. As some of the metrics may provide a lot of data it is useful to introduce a Time To Live (TTL) that allows metric data to expire if they are no longer needed.

Additionally we find several kinds of metrics that agglomerate data. We will need the typical statistical measurements such as count, sum, mean, median, standard deviation, frequency, distribution, and others. Together these metric properties can be used to provide (a) effective summary reports, (b) periodicity reports (c) real time information. Trough the introduction of stakeholders (see Section ??) some information may be restricted.

Examples of summary reports include walltime of servers used in a cluster as total over a particular time, the VM count, or the user count. Additional factors such as to which project the information is associated is important for project members or for those judging if the resources allocated for a project are justified. Center directors may need to report to their organization which scientific discipline have used the resource. These are just a very small but quite useful set of metrics that have had practical impact in existing academic cloud environments such as FutureGrid and FutureSystems. For summary reports we also found it useful to overlap multiple metrics into the same chart over a given period. This way we can showcase, contrast or verify trends between different metrics. To present the information the metrics may need to be able to be conveniently be displayed as charts, as time series data, in tables or json format or through RESTful service. Properties that are important to consider as part of metrics include scalability, overhead, availability, accuracy, security, agility, and the integration of some of the metrics into autoscaling services.

We provide a comprehensive list of useful metrics in Table 2.

3.1 Metric Keywords

To provide a starting point for discussions with other resource providers, we list a number of keywords that are associated with exemplary metrics that we are interested in deriving in more details while providing a subset of concrete metrics in Table ??.

```
o usage - runtime
  - VM
  - image
  - instance
  - regions
  - servers
  o network
    - public/private ip address
    - traffic
  o storage
    - block storage
    - objects
    - files
    - space
o performance
  - cpu usage (utilization)
  - network in/out bytes
  - disk read/write throughput
  - power consumption
  - memory usage
  - latency (e.g. network response or vm start)
o physical hardware monitoring
  - host contention
  - degraded hardware
  - outages
  - downtime
o automation
  - recovery
  - autoscaling
  - policy/threshold to take actions
```

Table 1: Selected Scientific impact metrics

Name	Description	Motivation
Scientific impact metrics		
Publication Count	Measuring how many publications are attributed as output while using the cloud resource	measures the general productivity
Citation Count	Measuring how many citations has been received as of the measuring time point	shows how the output really impacts the scientific community while the publications are being cited
H-index (person) [?]	A person has index h if h of his/her N_p papers have at least h citations each, and the other $(N_p h)$ papers have no more than h citations each.	measures both productivity and the generated impact
G-index (person) [?]	Given a set of articles ranked in decreasing order of the number of citations that they received, the g-index is the (unique) largest number such that the top g articles received (together) at least g^2 citations.	depends on the full citation count of very highly cited papers
H-index (entity)	While treating the evaluated entity as if it were a person, using the h-index that were originally defined for a person	measures both productivity and the generated impact
G-index (entity)	While treating the evaluated entity as if it were a person, using g-index that were originally defined for a person	measures both productivity and the generated impact
i10-index [?]	Number of publications cited as least 10 times	How many publications have achieved a certain level of impact (based on citation count)
ptrank-citation	Percentile rank of citation count among peers	To compare the studied entity's relative impact with the peers. The peers are identifie as publicatons that were published in same issue of the same journal for a publication that belongs to the studied entity (e.g., a journal, a Field of Science, the whole system, etc.)

  - live migration
o billing/charging/accounting
o monitoring service
  o errors
    - detect incidents
    - trace issues/ bugs
    - prevent long downtime
  - alert/notification (e.g. text, email)
  - diagnostics
  - security
  - capacity planning/prediction
  - trend changes
o audit/assurance
  - user behavior

```
o  VM
o  Usage
o  Failure
o  Prediction
```

```
Total User Count
Count over Group
Project, Center, Organization
Counts the active users in the cloud
```

ID realtime stakeholder = user, administrator, management, funder type = count, sum, distribution, ...

Hyungro: provide simple sceenshots of report figures related to the information listed in Fig 1a - 8a or are thes already displayed in the last section

Hyungro: provide a screenshot of the dashboard

Hyungro: provide a screenshot of a single page of the metric report with nice headlines and graphs

3.2 Sample Reports

Gregor: complete this section

Sample report information from FutureGrid is available as html [?] or as PDF [?]

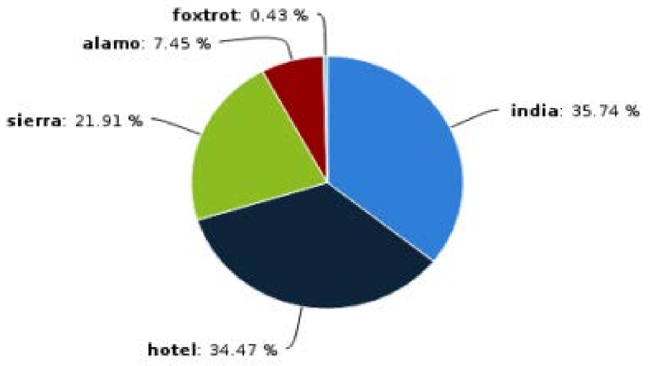


Figure 7: Active user count by host

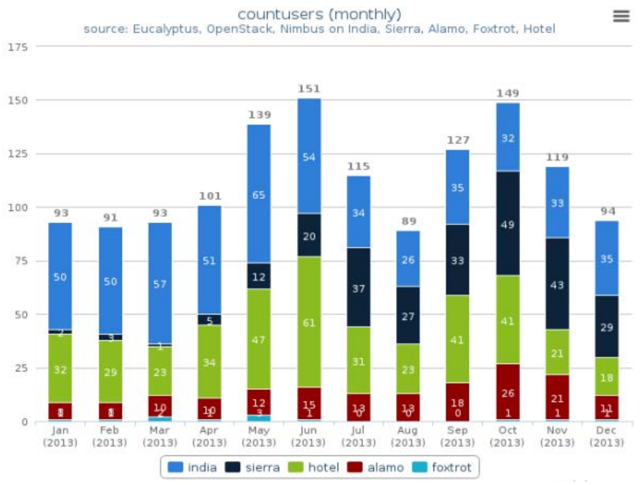


Figure 8: Active user count monthly

Table 2: Metrics Table

ID	Name	Stackeholder	Description	Motivation	Sample
User related metrics					
UC.1	User Count		This metric counts the active users for the cloud	User count is important for measuring the popularity of the cloud	
UC.2	Major Users		This metric shows heavy users in terms of consuming resources e.g. top 10 users	Top 10 Users is important to see who has the most impact on the shared resources	
UC.3	User Type		This metric shows a user type defined in an account system with a percentage of the total numbers. e.g. a project leader, an instructor, or a students.	User Type is important to see a proportion of users.	
UC.4	Repeat User		This metric shows the number of users who actively using services for a certain period. e.g. last 3/6/9 month: 16/32/24	Repeat User is important for measuring user activity for a certain period	
Virtual Machine related metrics					
VC.1	VM Count		This metric counts the launched VM instances on the cloud	VM Count is important for measuring the volume of requested instances	
VC.2	VM Count by Project, Leader, or Institution		This metric shows share of resource by group metrics such as project, leader or institution	Group usage is important for measuring group usage	
VC.3	Current running VMs or accessed users		This metric shows instant usage data to see current status	real time usage is important for checking peak or detecting unusual usage	
VCT.1	VM Creation Time		This metric shows the latency of creating a new instance to ensure fast creation	VM Create Time is important for measuring the latency	
Image related metrics					
IC.1	Image Type		This metric shows an image type registered in the cloud platforms with a percentage of the total number. e.g. ubuntu14.04, centOS 7, or Fedora 22	Image Type is important to see a proportion of images.	
Flavor related metrics					
FC.1	Flavor Type		This metric shows an flavor (instance) type registered in the cloud platforms with a percentage of the total number. e.g. m1.small, m1.medium, or m1.xlarge	Flavor Type is important to see a proportion of instance types.	
Storage related metrics					
SU.1	Disk Usage		This metric shows the size of Disks allocated to instances	Disk Usage is important for measuring system resource used	
SU.2	Object Usage		This metric shows the size of Object Storage allocated to instances	Object Usage is important for measuring system resource used	
SU.3	Block Usage		This metric shows the size of storage blocks mounted to instances	Block Usage is important for measuring system resource used	
Region related metrics					
RC.1	Location		This metric shows a geographical location of a user.	Location is important for resource availability to different regions.	
Server related metrics					
SC.1	Node Distribution		This metric shows a physical node distribution.	Node Distribution is important for load banacing.	
Network related metrics					
NL.1	Latency		This metric shows a network and application performance	Latency is important with acceptable and strict latency expectation	
NT.2	Network Throughput		This metric shows the actual amount of data delivered successfully	Network thoughput is important for measuring network performance	
NP.3	PublicIP Count		This metric shows availability of public IP addresses.	PublicIP Count is important for the number of available and free IP addresses.	
NP.3	PriveIP Count		This metric shows availability of private IP addresses.	PrivateIP Count is important for the number of available and free IP addresses.	

Table 3: Metrics Table

ID	Name	Stackeholder	Description	Motivation	Sample
Unordered metrics					
RS	Runtime Sum		This metric shows the total amount of runtime for launched instances	runtime by hour is important for measuring actual runtime of instances	
XX.X	vCPU Usage		This metric shows the number of vCPU cores allocated to instances	vCPU Usage is important for measuring system resource used	
XX.X	Memory Usage		This metric shows the size of Memories allocated to instances	Memory Usage is important for measuring system resource used	
XX.X	Node Distribution		This metric shows a proper balancing of physical compute nodes	Node Distribution is import for system load balance	
XX.X	Tenant Distribution (Std deviation)		This metric shows a proper balancing of resources per tenant. Useful to identify heavy tenants	Tenant Distribution is important to see users spread evenly across the nodes	
XX.X	Availability		This metric shows a percentage rate of available resources to accept a new request	Availability is important to provide cloud resources continuously	
XX.X	Scalability & Capacity		This metric shows an actual limit of a service or physical system in the cloud	Scalability and Capacity are important to measure IaaS performance	
XX.X	Power Consumption		This metric shows the amount of energy used in the cloud platform	Electricity is important for measuring actual cost by Kilo Watt per hour (KWh)	
XX.X	Throughput		This metric shows the performance of cloud services by measuring completed tasks, i.e. PaaS	Throughput is important for measuring service performance e.g. PaaS	
XX.X	CPU Speed (system performance)		This metric shows the performance of cloud resources by clock speed of a processor	CPU Clock speed is important to understand an actual speed of CPUs over different cloud platforms	
XX.X	Memory Speed (system performance)		This metric shows the performance of cloud resources by clock speed of a memory	Memory Clock speed is important to understand an actual speed of memories over different cloud platforms	
XX.X	Disk Speed (system performance)		This metric shows the performance of cloud resources by read/write speed of a disk including SSD	Disk speed is important to understand an actual speed of disks over different disk types	
Project metrics					
XX.X	Number of projects		x description	time periods: daily, monthly, qaurterly, yearly	
XX.X	Number of projects by discipline		x description	time periods: daily, monthly, qaurterly, yearly	
XX.X	Number of projects by organization		x description	time periods: daily, monthly, qaurterly, yearly	
XX.X	Count Technology		x description	time periods: monthly, qaurterly, yearly	
XX.X	Count Desired Technology		x description	x motivation	
XX.X	Comparision of desired Technologies		period: weekly, monthly, yearly. Type: Table, Frequency, Pie chart		
XX.X	Histogram desired X over Period		x description	x motivation	

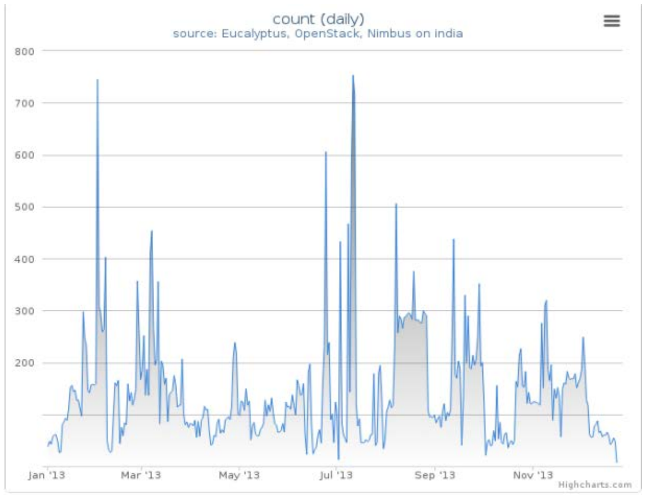


Figure 9: VM Count daily

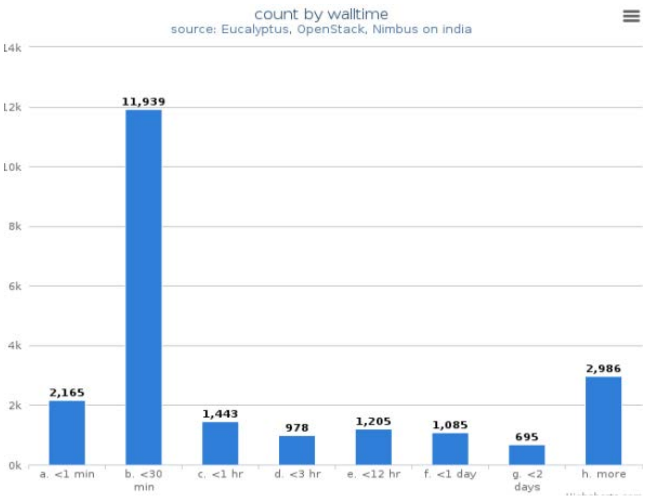
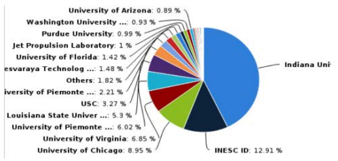
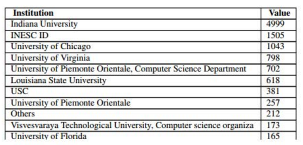


Figure 10: VM Count by runtime distribution





Institution	Value
Indiana University	4999
INESC ID	1505
University of Chicago	1043
University of Virginia	798
University of Piemonte Orientale, Computer Science Department	702
Louisiana State University	618
USC	381
University of Piemonte Orientale	257
Others	212
Visvesvaraya Technological University, Computer science organiza	173
University of Florida	165

Figure 11: Usage by Institution

3.3 Error and Alert Reporting

Cloudmesh Metrics reads system and application logs to generate error reports. If there are error messages stored in the database, Cloudmesh Metrics can lookup the database to collect the messages. Error information collected are stored in the Metrics database.

[?]

4. MONITORING TOOLS AND SERVICES

4.1 IaaS

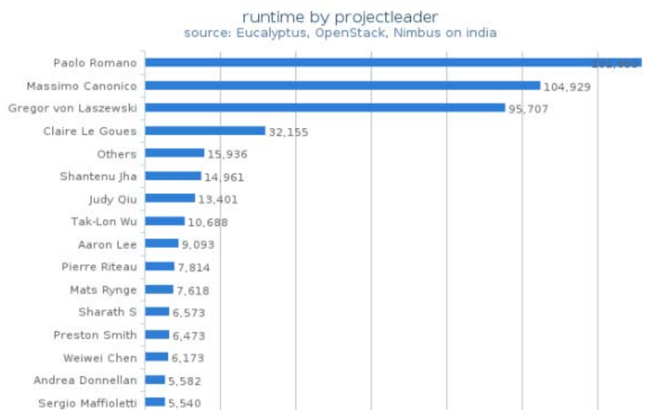


Figure 12: Runtime (Hour) by Project Leader

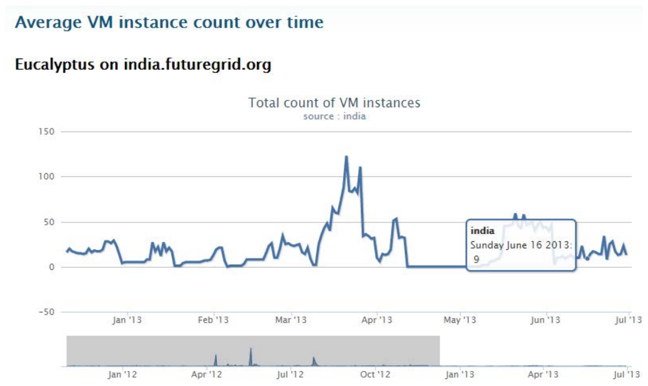


Figure 13: Real-Time Usage of VM instances

4.1.1 OpenStack

Openstacks Ceilometer provides a service for measuring usage data from openstack components to achieve monitoring and metering purposes. Ceilometer acquires measurements across current OpenStack components such as Nova (compute), Network, and Storage (swift). Next to an API it also provides a command line tools to retrieve usage statistics. However in deployments such commands are typically limited by policy to system administrators. By default it provides hourly information for compute utilization; instance type; availability zone; cpu core; memory size; nova volume block device type and availability zone; Network; data transfer (in / out), availability zone; external floating ip; Storage (Swift); disk size used; and data in/out.

There are four basic components to Ceilometer:

Agents. Agents run on each compute node and polls for resources utilization statistics.

Collector: A collector runs on management servers to manage the message queues for data coming from the agent. Metering data are stored to the openstack data store and a notification message are delivered to the Openstack messaging bus once they are processed.

Data store: is a place of collected data. It provides interaction with the collector and a api server.

API server: runs on management servers to provide statistics about the measured data.

Table 4: Error Metrics

Metric	Group	Unit	Purpose	Sample
Error Rate	host, project, or term	Percentile	Measure Availability	2% errors in i57 host, 4% errors in April 2015
Error Count	host, project, or term	Count	Identify Issues	399 User Errors (5xx), 133 errors of Instance typeś disk is too small for requested image (501)
Debug	host, project, or term	Count	Troubleshooting	133 times failed of _build_instance() function in manager.py with error code 5xx
Usage Alert	VM, CPU, RAM or Disk	Count or Size	Notify Limit	168 of 180 VMs used (93% used)
Trace Resource	VM, CPU, RAM or Disk	Count or Size	Inform status	120/168 free/avail disks

Table 5: Basic Concept of Metric

Name	Unit	Type of Value	Refresh Interval (Measuring time)
Runtime	hour, min, sec	cumulative, aggregation	event-triggered
running jobs	count	delta, (change from the previous value)	time-triggered e.g. 5 secs, 5 mins
CPU, Memory, Load	Percentage	gauge, (standalone value relating only to the current duration)	time-triggered e.g. 5 secs, 5 mins

Production scale metering is estimated to have 386 writes per second and 33,360,480 events a day, which would require 239 Gb of volume for storing statistics per month [?].

The OpenStack Orchestration program, Heat, provides an autoscaling service while leveraging Ceilometer. It is similar to Amazons CloudFormation. This integration of Heat and Ceilometer allows users to develop services that provide better resource utilization. This is similar to the combination of the AWS Auto Scaling and AWS CloudWatch to provide the autoscaling service based on monitoring values [?].

In addiction to ceilometer the command line tools provided by OpenStack Compute (Nova) offer elementary usage statistics, but also inform users about quota and API call limitations. For more detailed information about resource usage users will need to use the information exposed with ceilometer. Examples of the information includes host information as depicted in Figure 15 and usage data as depicted in Figure 15.

```
$ nova host-describe i40
+--------+------------+-----+-----------+---------+
|  HOST  | PROJECT    | cpu | memory_mb | disk_gb |
+--------+------------+-----+-----------+---------+
| i40    | (total)    | 8   | 24100     | 2698    |
| i40    | (used_max) | 10  | 19456     | 180     |
| i40    | (used_now) | 10  | 18944     | 180     |
| i40    | project1   | 1   | 2048      | 20      |
| i40    | project2   | 8   | 16384     | 160     |
| i40    | project3   | 1   | 512       | 0       |
+--------+------------+-----+-----------+---------+
```

Figure 14: Display a summary of resource usage of the devstack-grizzly host

```
$ nova usage-list
Usage from 2014-02-14 to 2014-03-15:
+--------+---------+-------------+---------+-----------+
| Tenant | Instan- | RAM         | CPU     | Disk      |
| ID     | ces     | MB-Hours    | Hours   | GB-Hours  |
+--------+---------+-------------+---------+-----------+
| user1  | 17      |  6840394.43 | 3340.04 |  66800.73 |
| user2  | 17      |   185683.06 |   90.67 |   1813.31 |
| user3  | 1       |   932256.36 |  455.20 |   9104.07 |
| user4  | 26      |  4947215.08 | 2415.63 |  48312.65 |
| user5  | 5       | 18644854.23 | 9103.93 | 182078.65 |
+--------+---------+-------------+---------+-----------+
```

Figure 15: Summary statistics for tenants

Internally usage data for Ceilometer and the nova command line tools is provided by OpenStack Notification System. The notification system can be configured to emit events either through nova's logging facility, or send them to a series of AMQP queues (one per notification priority). System usages are emitted as notification events with the INFO priority. Different types of usage events are distinguished via the notifications `event_type`, which is a hierarchical dotted string such as `compute.instance.create`, which allows usages to be easily grouped for aggregation. Usage notifications can be immediate (created when a specific increment of usage occurs, such as creation of an instance) or periodic (generated by a periodic task, like a cron job) and cover usage for a certain time period. A notification includes attributes such as id, priority, event type, and timestamp associated with a notification data payload as a json-formatted key-value pair [?]. Through this set of information custom parsers and information queries can be generated. However it does require policies to be set up to gain access to this information.

4.1.2 Eucalyptus

Eucalyptus provides a to several Amazon services compatible private cloud. It offers resource usage information through external monitoring tools such as Nagios and Ganglia. To enhance system management, Eucalyptus provides nowadays summary reports about resource allocation and status. There are commands line tools for generating reports for eucalyptus cloud that start with eureport- in the Cloud Controller (CLC) and eucadw- in the data warehouse. The reports provide usage data for understanding how cloud resources are utilized and being used via simple command line tools. eureport-generate-report is a main command to get access usage data. Various type of resources can be measured such as elastic-ip, instance, s3, snapshot, and volume when eureport-generate-report is ran with a report type option. The Eucalyptus data warehouse is a place to keep all usage data coming from CLC. External programs can get access to the usage data from the data warehouse instead of CLC directly. It may reduce impact of pulling usage infor-

Table 6: System Performance Metric (OS Level)

Metric	Description	Type	Level	Unit	Example
CPU Utilization	Percentages of total CPU time	System Monitoring	Operating System	Percentage	vmstat
I/O Read	Read operations on disks			Count or Bytes	iostat
I/O Write	Write operations on disks			Count or Bytes	
Network In	Received bytes to network interfaces			Bytes	
Network Out	Sent bytes from network interfaces			Bytes	

Table 7: Network Monitoring Features (source from wikipedia)

Metric (legend)	Description	Ganglia	Nagios
Trending	Provides trending of network data over time	Yes	Yes
Trend Prediction	The software features algorithms designed to predict future network statistics	No	No
Auto Discovery	The software automatically discovers hosts or network devices it is connected to	Via gmond check in	Plugin
Agentless	The product does not rely on a software agent that must run on hosts it is monitoring so that data can be pushed back to a central server.	No	Supported
SNMP	able to retrieve and report on SNMP statistics	Plugin	Plugin
Syslog	Able to receive and report on Syslogs	No	Plugin

mation from cloud when it performs its cloud duties [?]. Previously Eucalyptus did not have a strong monitoring framework and as part of FutureGrid we developed a system that parses log file events on the management node to provide them. In future we need to evaluate if the existing Eucaluptus monitoring features could replace this framework. However, as the usage demand of Eucalyptus has almost diminished, we have in FutureSystem discontinued the use of Eucalyptus. University of Buffalo is in the process of deploying a Eucalyptus cloud and we hope the integration of Eucalyptus data into a heterogeneous metrics framework can be continued at that time. At this time we are not aware of any other new Eucalyptus cloud deployments.

4.1.3 Azure

Microsoft Windows Azure is a cloud computing platform used to build, host and scale web applications through Microsoft data centers [?]. The platform contains various on-demand services hosted in Microsoft data centers. These services are provided through three products.

Windows Azure: an operating system that provides scalable compute and storage facilities.

SQL Azure: a cloud based, scale out version of SQL server.

Windows Azure AppFabric: a collection of services supporting applications both in the cloud and on premise.

The System Center Monitoring Pack for Windows Azure application is according to Microsoft the most cost effective and flexible platform for managing traditional data centers, private and public clouds, and client computers and devices [?]. It provides monitoring of availability and performance for Windows Azure applications. It provides a unified management platform where multiple hypervisors, physical resources, and applications can be managed in a single offering. From a single console view, the IT assets like network, storage and compute can be organized into a hybrid cloud model spanning the private cloud and public cloud services. The monitoring pack runs on a specified agent and uses Windows API.s to remotely discover and collect information about a specified Windows Azure application. By default, the monitoring is not enabled. Therefore, the discovery must be configured by using the Windows Azure Application monitoring template for each Windows Azure Application to be monitored. The following functionalities are provided by the Monitoring Pack for Windows Azure Applications:

- Discovers Windows Azure applications.
- Provides status of each role instance.
- Collects and monitors performance information.
- Collects and monitors Windows events.
- Collects and monitors the .NET Framework trace messages from each role instance.
- Grooms performance, event, and the .NET Framework trace data from Windows Azure storage account.
- Changes the number of role instances.

Implementing monitoring means, launching the diagnostic instance and this instance will collect the data and at the interval user wants. The collected data will be copied to an Azure Table to record information for performance counters and windows event logs. The performance monitoring can be enabled by direct implementation or using tools such as powershell cmdlets for Windows Azure [?] or the Azure Diagnostics Manager 2 from Cerebrata [?].

By using these tools one instance of Windows Azure is configured to collect some performance counters without modifying the application code. The performance data will be collected by the Azure Diagnostic Monitor and moved at the interval user specified. Hence the user can use this diagnostic data for debugging and troubleshooting, measuring performance, monitoring resource usage, traffic analysis and

Table 8: Metrics for Cloud Computing

Metric	Cloud Platform	Purpose	Related Service
CPU Utilization	IaaS, PaaS	Performance	Scale Up/Down
Task completion time	PaaS	Performance	
Number of VMs	IaaS	Traffic	Scale Up/Down
VM sizes (flavors)	IaaS	Capacity	

Table 9: System Monitoring on Cloud platforms

Metric	AWS	GCE	Azure	OpenStack	HP Cloud (Eucalyptus)	Nimbus
CPU Utilization	Yes	TBD	TBD	TBD	TBD	TBD
I/O Read	Yes	TBD	TBD	TBD	TBD	TBD
I/O Write	Yes	TBD	TBD	TBD	TBD	TBD
Network In	Yes	TBD	TBD	TBD	TBD	TBD

capacity planning, and auditing. Diagnostic data is not permanently stored unless user transfers the data to the Windows Azure storage emulator or to Windows Azure storage. After the data is transferred to storage it can be viewed with one of several available tools. To collect Windows Event logs in a Windows Azure application, the Event logs data source must be configured. Access is controlled by authentication access control. The monitoring data can be visualized using System Center Operation Manager Console. From Operation Manager, user can create custom dashboard or publish graphs on SharePoint to people who do not have the SCOM console.

4.1.4 Amazon

Amazon CloudWatch.

Amazon CloudWatch (ACW) [?] monitors Amazon Web Services (AWS) resources and the applications users run on AWS in real-time. ACW is a metrics repository. The AWS products ingest metrics into the repository, and users retrieve statistics based on those metrics. Metrics are time-ordered sets of data points, are isolated from one another in different namespaces so that metrics from different applications are not mistakenly aggregated into the same statistics. Users retrieve statistics about those data points as an ordered set of time-series data. Over the time value is important for metrics since it contains historical changes in it. An API is provided to interface programmatically.

CloudWatch allows notification to alert users and auto scaling (automatically make changes) to the resources you are monitoring based on rules that you define. Hence, CloudWatch can manage thresholds to send a notification to users via email or text messages, and even more, apply changes with a pre-defined settings such as increasing virtual instances or diminishing. System-wide visibility into resource utilization, application performance, and operational health can be provided on a users services. However insight into the underlaying IaaS framework is not sufficiently provided as is tha case in academic private clouds.

4.1.5 Rackspace Cloud Monitoring

Rackspace Cloud Monitoring is an API driven monitoring system which allows administrators to use or create APIs depending on their needs which can send notifications to any device including mobile devices. This allows administrators to be on top of their Rackspace-hosted infrastructure which includes websites, protocols, and ports.

4.1.6 Google Cloud

Google launched Cloud Monitoring in 2014 after the acquisition of Stackdriver, a software company that developed monitoring services in the cloud. Google Cloud Monitoring (GCM) provides performance data of cloud resources and sends notifications if issues occur based on a user defined alerting policy. Monitoring is available in Google App Engine, Compute Engine, Pub Sub, Cloud SQL and Cloud Storage with health check, log tracing on the web-based Cloud Monitoring Console according to the Cloud Monitoring website [?]. Additional metrics such as memory or disk usage can be collected by the Monitoring Agent which requires to be installed in a VM instance. Google offers Monitoring for free during its Beta release.

4.2 HPC

Monitoring in high-performance computing has similar factors as described earlier. A number of existing tools support monitoring and metrics for HPC computing. Some of them are directly build into the queuing system. We describe a selected number of tools and services next.

4.2.1 XDMoD

XDMoD (XSEDE Metrics on Demand) [?, ?, ?] is developed as a successor to UBMoD (University Buffalo Metrics on Demand) while trageting firstly the XSEDE resource providers. Most recently an open source version can however be deployed at non XSEDE based resource providers. It is a tools for collecting and mining statistical data from cluster resource managers such as Torque[?], Maui[?], Moab[?], OpenPBS [?], SGE[?] and Slurm [?] commonly found in high-performance computing environments. XDMod has three fundamental components: a metrics repository (e.g. a Data Warehouse), a RESTful API, and a web-based application (Portal). Its web graphical user interface provides rich set of tools to expose various statistics with different type of charts and tables while communication through a RESTful API [?, ?].

4.2.2 *Nagios*

Nagios [?] is a web based Linux monitoring systems allowing to monitor availability and response time of network services, usage of system resources like CPU load, RAM allocation etc., number of logged in users and so on. The main Nagios server collects information from Linux, BSD, Windows hosts or Cisco devices through Nagios clients (agents), and records their states. A web interface is provided to expose the information to authorized users. Nagios generates a notification in case of any outage detected or any anomaly through wide range of alert methods. They can be forwarded via e-mail, sms, chat messages and phone call notifications. As Nagios monitors states but it typically does not expose graphs such as network interface usage.

4.2.3 *Cacti*

Cacti is a network monitoring tool using the simple network management protocol (SNMP). It is similar to other tools such as Nagios [?] and Ganglia. It uses RRDtool [?] as a DBMS to provide a round robin visualization of the monitored information. Since it supports polling, monitoring data via shell scripts, php, or c-based executable is possible. t can be extended to measure other resources besides network traffic.

4.2.4 *Zabbix*

Zabbix is an open source monitoring service for networks and servers while accessing information via a number of standard protocols such as SNMP [?], IPMI [?], TCP, [?] and JMX [?]. The centralized server of Zabbix collects monitoring data through several Zabbix agents installed on desirable hosts and other servers. The information is store in a database and displayed through a web interface that can create reports on-demand. The agents To support cloud monitoring, Zabbix active agent auto-registration enables monitoring cloud instances on IaaS such as Amazon Web services, and OpenStack.

4.2.5 *Zenoss*

Zenoss (Zenoss Core) provides a unified resource management service which manages applications, networks, servers, and storage. It allows to monitor physical or virtual systems including the public, private and hybrid clouds. It is built on top of the Zope object-oriented web application server, and using RRDtool with MySQ. It can collect information using SNMP, SSH, WMI and log files e.g. syslog. Zenoss provides additional features with other monitoring mechanisms such as Perfmon, JMX, and VM API (e.g. VMware API) in the enterprise version. A key feature of Zenoss is model-driven monitoring allowing to automatically discover, configure and monitor services. As a communication tool, Zenoss utilizes the Twisted Perspective Broker (PB) [?] instead of AMQP typed messaging system such as RabbitMQ. PB is an an asynchronous, event-driven with co-operative multi-tasking which is a deferred object.

4.2.6 *Inca*

Inca is monitoring software for watching user-level activities and measuring performance on a Grid including detecting errors [?]. The executable program, Inca reporter, collects performance data from a system and Inca web interface provides detailed information about the measured data. XSEDE uses Inca for monitoring GRAM usage, CA and CRL validity and resource registration in MDS according to the project description page [?]. FutureGrid project used Inca for measuring performance of FutureGrid software and hardware with 196 Inca reporters for testing FutureGrid services and clusters citeinca-futuregrid. Inca is used on the XSEDE resources Keenland [?][?] and Darter [?].

4.2.7 *Ipm*

Ipm [?, ?, ?] ...

Hyungro: Describe what ipm is

Ipm is used on the XSEDE resource Stampede [?]. [?]

4.2.8 *PAPI*

Hyungro: describe what papi is

PAPI is used obn the XSEDE resource Stampede [?].

4.2.9 *Gold*

There is a small number of accounting software to provide metered resource utilization in open source cloud platforms. Those tools such as Gold accounting Manager [?, ?] are very simple and are supposed to support system administrators not cloud users. However, Gold is no longer supported by the developers.

4.2.10 *XRAS*

Hyungro: describe what xras is

[?]

5. CLOUDMESH METRICS SERVICES AND FRAMEWORK

Cloudmesh Metrics extends our cloudmesh framework [?] by integrating a metrics framework for monitoring essential IaaS metrics. It can handle handling multiple heterogeneous cloud platforms, to provide measurement of usage data about cloud resources. It collects usage data directly from various cloud management services and integrates them into a single report. This framework was extensively used in FutureGrid [?, ?] and its use is continued in FutureSystems [?]. It has served as integrator to three different IaaS frameworks, namely, Eucalyptus, OpenStack, and Nimbus. It has served to provide integrated information of at least eight different cloud deployments on various hosts in FutureGrid and FutureSystems. The native usage data from different clouds are integrated into the Metrics database. Analysis of this data can be integrated into different report formats including PDF, and HTML. An API and a command line interface is available to customize reports while exposing them also through a REST interface. A Web interface provides a number of predesigned reports. Interactivity to min the graphs [?] is provided through java script enabled charts or tables.

5.1 Design

The design of the cloudmesh metrics framework allows the mashup of information from various data sources including cloud management databases and log files. This mashup provides us with a rich data set supporting sophisticated analysis that may spawn multiple information sources. The system has been used in FutureGrid to support metric information from multiple IaaS service providers from various

FutureGrid partner sites, for more than 400 projects and over 2800 members as of end of 2014 [**?**].

Figure 16 shows a layered architectural view of the Cloudmesh Metrics service and how it interfaces with various IaaS frameworks.

Cloudmesh Metrics consists of four components: (1) Metric Collector: a measuring tool of resource allocation, (2) Cloudmesh Metric Shell providing a CLI tool to define metrics and collect usage data, (3) Cloudmesh Metric Portal and Report providing a visualization tool to provide graphical representative of the data, and (4) a Web Service APIs to support other applications e.g. scheduling and dynamic provisioning.

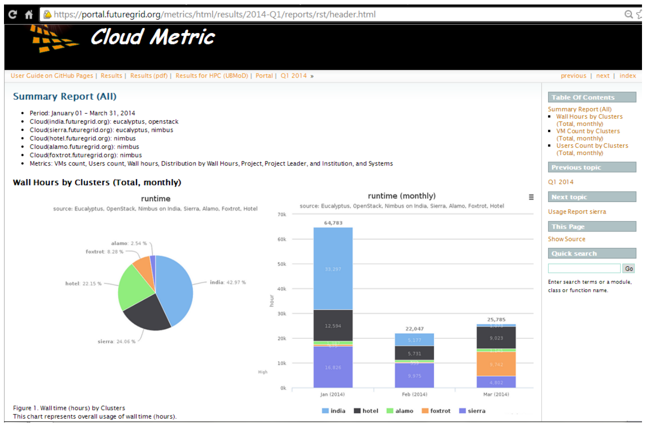
The Metric Collector enables collecting usage data allocated from IaaS cloud platforms such as Eucalyptus, Nimbus, and OpenStack including HPC from TORQUE. A log parser extracts data from log messages of IaaS platforms and transforms them into metrics. The metrics are then stored into a database that allows exposing the data as JSON objects via API calls. The collecor enables also real-time monitoring and statistics. Data can be integrated from OpenStack MySQL databases or Nimbus sqlite3 to the unified database. Fro Eucalyptus we obtained most of the information through its log files as at the time Eucalyptus did not provide a sophisticated monitoring or metrics framework beyond the available logging facility. Account information such as a user name and a project associated with are also imported to the main database so the overall usage data can be viewed for vm instances and jobs launched on FutureGrid resources via LDAP.

To keep the load on the original services minimal, we pull data on regular intervals from these production systems. Other mechanisms such as access to replicated databases could easily be supported.

5.2 Selected Analysis with Cloudmesh Metrics

To showcase the usefulness of the system we present some simple examples from our operation of FutureGrid and FutureSystems spawning multiple years of production deployment. We early found that a simple summary view provides valuable input to management and funders. Furthermore the automatically generated reports at predefined intervals serve the required quarterly reporting. An examples of such summaries are provided in Figure 17. Real time information about virtual machine usage is depicted in **??**.

Based on the observation on FutureGrid, there is a different pattern between a research project and class work when they acquire cloud resources. Resource allocation of academic coursework shows time dependent request patterns. It shows a surge when there is a class, a lab session, and a project. For example, the undergraduate course for Distributed Systems at Indiana University introduced IaaS in the class and used the IaaS platform for a class project. Figure 19 shows a spike in the class and variability until the project due. Additionally, we have been able to expose to the class lead that despite students attending this class not all students have actually logged into the systems or created VMs. Such information may provide valuable input to the teacher in order to verify group work or avoid academic dishonesty. It also contradicted the demand for the faculty member to "reserve" for the entire semester a substantial number of compute resources so they can be exclusively used by the class. Although we observe a spike,


https://portal.futuregrid.org/metrics/html/results/2014-Q1/reports/rst/header.html

Cloud Metric

User Guide on GitHub Pages | Results | Results (pdf) | Results for HPC (UBMoD) | Portal | Q1 2014 » previous | next | index

Summary Report (All)

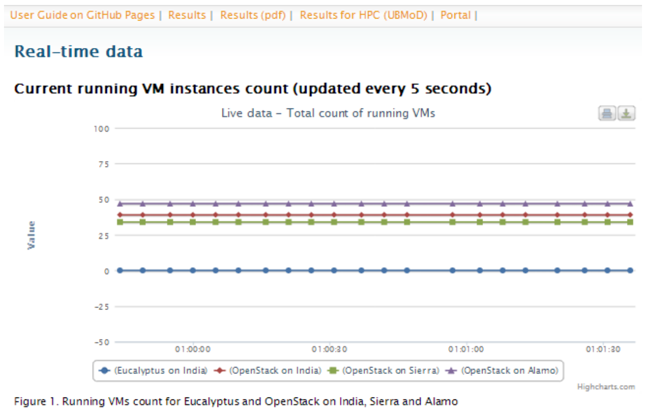
- Period: January 01 – March 31, 2014
- Cloud(india.futuregrid.org): eucalyptus, openstack
- Cloud(sierra.futuregrid.org): eucalyptus, nimbus
- Cloud(hotel.futuregrid.org): nimbus
- Cloud(alamo.futuregrid.org): nimbus
- Cloud(foxtrot.futuregrid.org): nimbus
- Metrics: VMs count, Users count, Wall hours, Distribution by Wall Hours, Project, Project Leader, and Institution, and Systems

Wall Hours by Clusters (Total, monthly)

Enter search terms or a module, class or function name.

Figure 1. Wall time (hours) by Clusters
This chart represents overall usage of wall time (hours).

Figure 17: Cloud Metrics Portal pie chart and stacked bar chart represent monthly usage of FutureGrid resources. Summary of regional clusters and different IaaS cloud platforms are displayed with various charts and different terms including monthly, quarterly and yearly.


User Guide on GitHub Pages | Results | Results (pdf) | Results for HPC (UBMoD) | Portal |

Real–time data

Current running VM instances count (updated every 5 seconds)

Figure 1. Running VMs count for Eucalyptus and OpenStack on India, Sierra and Alamo

Figure 18: Real-time usage report on Cloud Metrics Portal: In every 5 seconds, the line chart is updated with the number of working virtual machine instances.

the available resources in the overall cloud would have been sufficient to satisfy the user demand.

While we see time dependent use in classes, many research project show in contrast VM instances requests on a more regular basis. An example is found in the Next Generation Sequencing (NGS) in the cloud project on FutureGrid shows relatively consistent resource allocation requested in Figure 20. With a certain period of time, vm instances of this project have been launched without unplanned spike requests. These two examples show different patterns for deploying resources but both cases have a factor to predict loads. The class schedule and the monitoring data for applications can be used to measure the amount of resources and identify incoming requests. While having this information available through our project registration makes it possible to predict in future similar behavior. Hence understanding these patterns is important to bring cost effectiveness over on-demand allocation.

We can furthermore analyse the data for the class usage

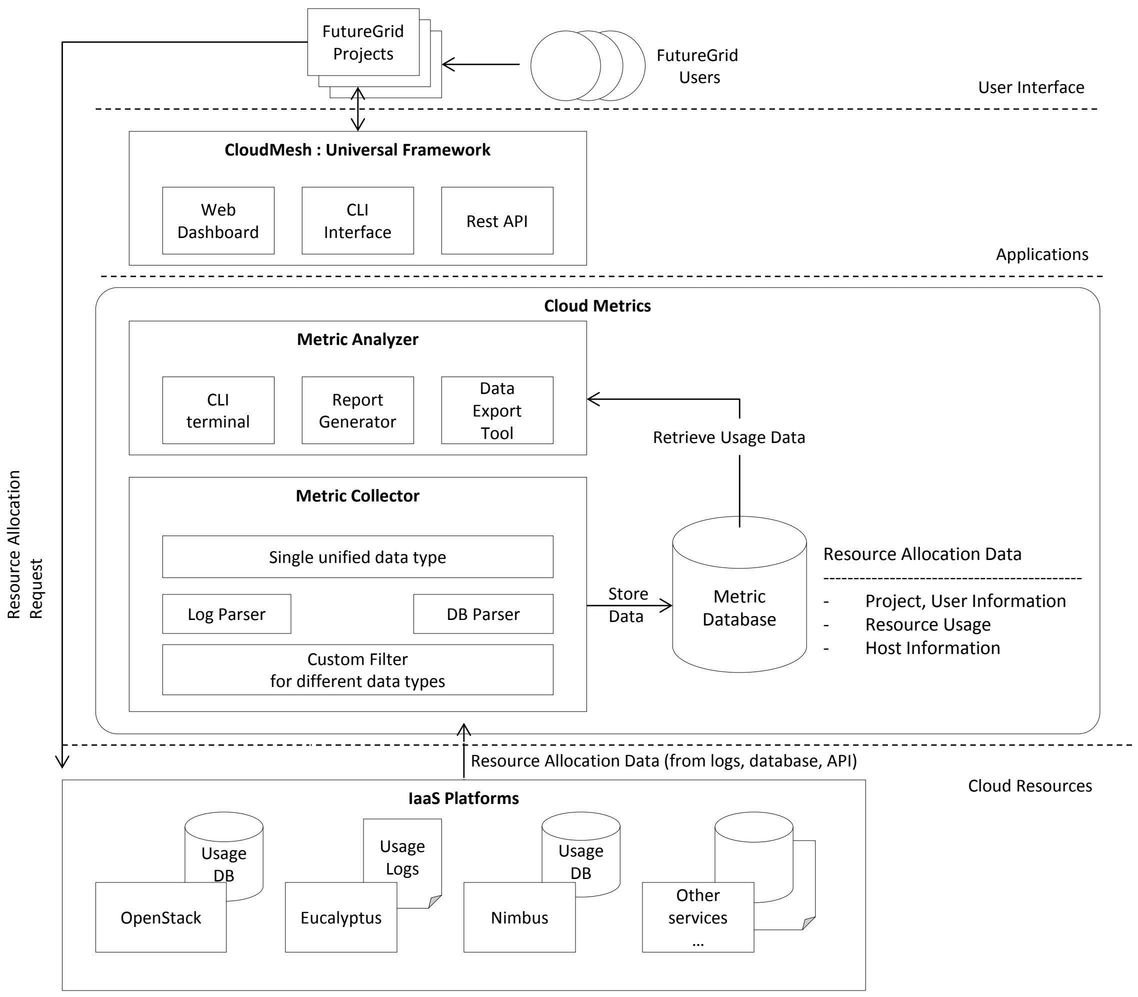


Figure 16: Overview of Cloud Metrics

Hyungro: we need to redraw, can this be dne in ppt or omni graffel

as shown in the Gantt chart Figure 21. Here we display the duration of the VMs as used in the class. At the beginning of the class, the gaps between the start and completed dates of the vm instances are small but a large number of instances are initiated. Once the class is became operative, the runtime of vm instances is getting longer and a less number of instances are requested compared to the beginning. This observation tells us that academic projects require training sessions using short running VMs at the beginning of a project to get familiar with using infrastructure and to prepare environments by installing software and datasets.

Another observation is that this particular class had a quite high usage of VMs for administrative purposes by the instructor. Figure 22 describes that instructors consumed a large number of vCPU cores before class starts and small tests just before class projects. It indicates that the preparation of courses require extensive load testing on cloud resources to estimate compute capacity needed for applications.

For this class 25 hosts, with 216 vCPUs and 600GB memories were reserved as to satisfy the request of accessing large virtual instances. In Figure 23 we demonstrate that the dedicated resources were being underutilized most time although the high volume requests had been made including a 273% overutilization on October 21th for testing and preparing. However, if the resources would have not been separated as part of its own cloud it would have been possible to allow resource aggregation and policy definitions that allow autoscaling in support of this class.

High Performance Computing (HPC) has been used to support parallel data processing of big data. In Figure 24 we see that in addition to IaaS big data projects have also requested HPC and other services. Additionally we can see that the initial dominant usage of Eucalyptus and Nimbus as IaaS has been replaced with OpenStack. Toghether this data shows it is desirable to provide a hybrid cloud that integrates HPC and IaaS service to scientists and researchers.

Another important result was that the number of servers per job requested as part of HPC usage increased. We found that for HPC, 64 and 128 CPU cores per job are most popular job sizes in FutureGrid HPC in 2013 (See Figure 25). 24% and 14% of total wall time are for 64 and 128 cpu jobs.

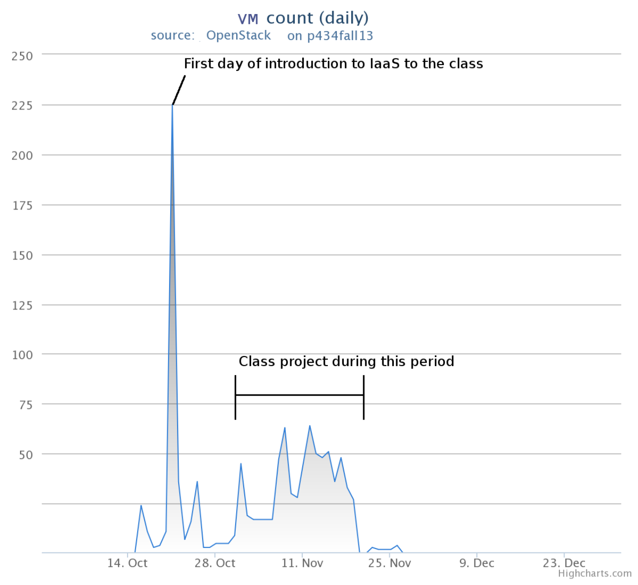


Figure 19: IaaS Usage data for the Distributed System class at Indiana University*

An extra large jobs (i.e. 512 CPU cores) has been intensively used in the last year. Compared to the previous year 2012, the request has been increased about 350%. In early stage of FutureGrid between 2010 and 2011, tiny CPU jobs have been requested many times but in 2013, two thirds jobs are using more than 64 CPU cores. This shows the user community in FuturGrid matured using more servers at a time than before.

CLearly we see from our examples that valuable information can be derived already with the cloudmesh metrics framework. The ability to have access to standard reports makes reporting and oversight easier. Furthermore data we collected could be used to convince those that are used to exclusive use of resources in favor of shared cloud resources.

6. RELATED WORK

This section will contain some relevant related work. We have included a small set of references that may be helpful to establish this section. Right now we provide a list of unordered refernces

Nist [?] Blueflood [?] CM-measurement facets for cloud performance [?] Online detection of utility cloud anomalies using metric distributions [?] M4Cloud-Generic Application Level Monitoring for Resource-shared Cloud Environments. [?] Design of a Dynamic Provisioning System for a Federated Cloud and Bare-metal Environment [?] [?] [?] [?] [?] [?] [?] [?] [?] [?] [?] [?] [?] [?] [?] [?] [?] [?]

6.1 Surveys and Taxonomies

This section will include a small survey of paper that provide them selfs surveys to cloud metrics.

[?] [?] [?]
[?] [?] [?] [?] [?] [?] [?] [?] [?] [?] [?] [?]
RightScale [?]

7. CONCLUSION

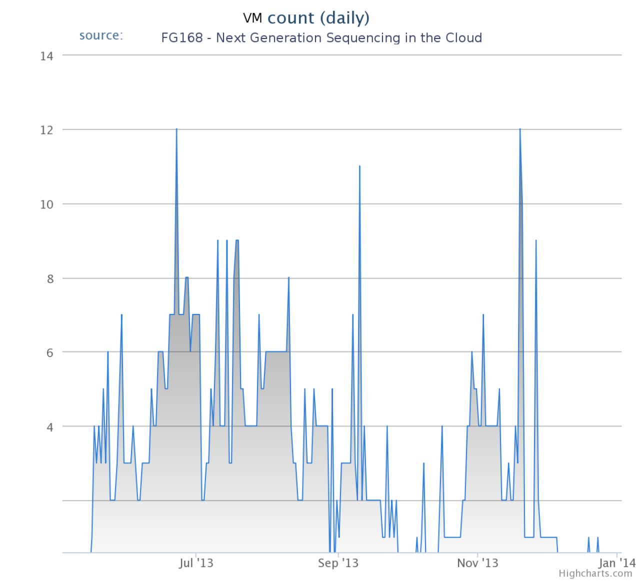


Figure 20: VM count for Next Generation Sequencing (NGS) in the cloud project

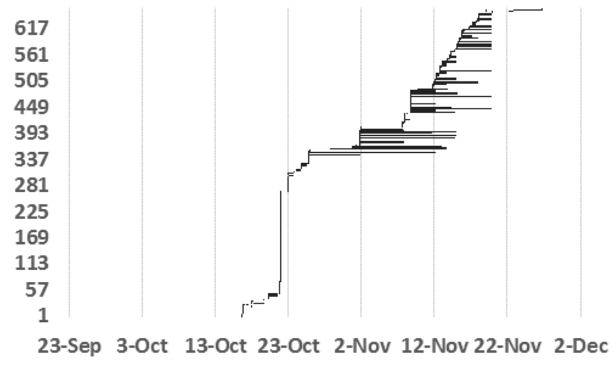


Figure 21: Timeline for VM walltime

write conclusion

8. ACKNOWLEDGEMENT

This material based upon work is partially supported in part by the National Science Foundation under Grant No. 1445806, 1445806, and 0910812.

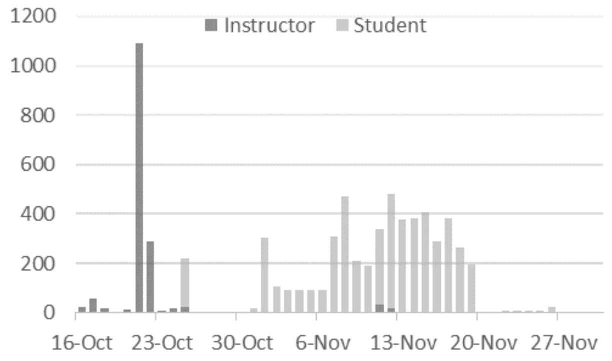


Figure 22: Usage between instructors and students for vCPU cores

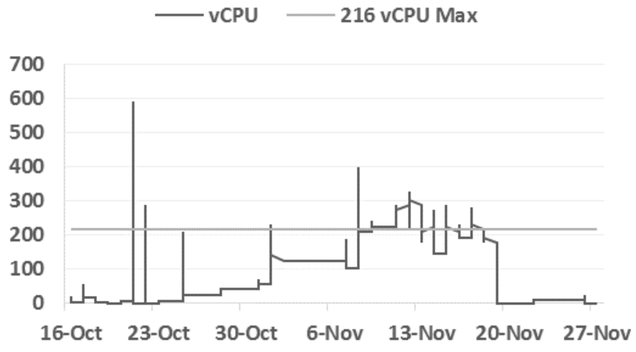


Figure 23: vCPU Utilization (approximation per hour)

Table 10: 1st Category in XDMoD

HPC Category	Compatibility with IaaS	Description
Job	VM	TBD
Allocations	Project(Tenant)	A funded project that is allowed to run jobs on resources
Accounts	Accounts	TBD
Requests	Requests	The number of service registration
Performance	Performance	TBD
SUPREMM	-	TBD

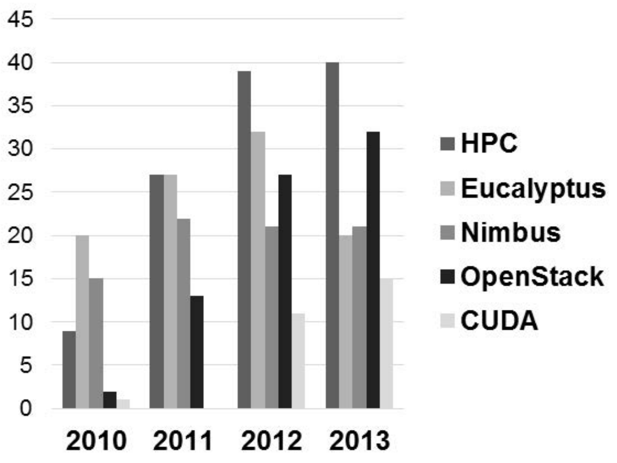
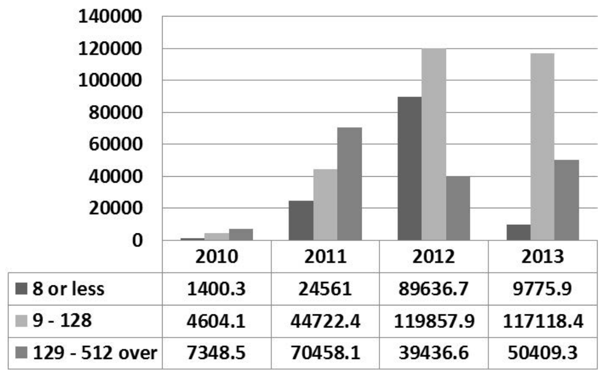


Figure 24: Service changes for along with Big Data between 2010 and 2013



	2010	2011	2012	2013
▨ 8 or less	1400.3	24561	89636.7	9775.9
▨ 9 - 128	4604.1	44722.4	119857.9	117118.4
▨ 129 - 512 over	7348.5	70458.1	39436.6	50409.3

Figure 25: Annual Wall Time Changes for Job Size in HPC between 2010 and 2013

Table 11: 2nd Category in XDMoD

Metric	Compatibility with IaaS	Description
Allocation Usage Rate	TBD	The rate of XSE
CPU Hours: Per Job	vCPU Hours: Per VM	TBD
CPU Hours: Total	TBD	TBD
Job Size: Max	VM Size: xlarge	TBD
Job Size: Min	VM Size: micro	TBD
Job Size: Normalized	TBD	TBD
Job Size: Per Job	TBD	TBD
Job Size: Weighted By XD SUs	TBD	TBD
NUs Charged: Per Job	TBD	TBD
NUs Charged: Total	TBD	TBD
Node Hours: Per Job	TBD	TBD
Node Hours: Total	TBD	TBD
Number of Allocations: Active	TBD	TBD
Number of Institutions: Active	TBD	TBD
Number of Jobs Ended	Number of VMs Terminated	TBD
Number of Jobs Running	TBD	TBD
Number of Jobs Started	TBD	TBD
Number of Jobs Submitted	TBD	TBD
Number of Jobs via Gateway	TBD	TBD
Number of PIs: Active	TBD	TBD
Number of Resources: Active	TBD	TBD
Number of Users: Active	TBD	TBD
User Expansion Factor	TBD	TBD
Wait Hours: Per Job	no wait	TBD
Wait Hours: Total	TBD	TBD
Wall Hours: Per Job	Running Hours: Per VM	TBD
Wall Hours: Total	TBD	TBD
XD SUs Charged: Per Job	TBD	TBD
XD SUs Charged: Total	TBD	TBD
XSEDE Utilization	TBD	TBD

Table 12: 2nd Category II in XDMoD

Metric	Description
Number of User Accounts: Closed	TBD
Number of User Accounts: Created	TBD
Number of User Accounts: Open	TBD

Table 13: 2nd Category III in XDMoD

Metric	Description
Avg IO Performance	TBD
Avg Memory Performance	TBD
Avg Network Performance	TBD
Avg System Performance	TBD
CPU:Graph500 - Performance	TBD
CPU:OSJitter - Inv Mean Noise	TBD
CPUIO:NWCHEM - Performance	TBD
CPUNET:HPCC - DGEMM	TBD
CPUNET:HPCC - FFTW	TBD
CPUNET:HPCC - LINPACK	TBD
CPUNET:NPB - BT	TBD
CPUNET:NPB - CG	TBD
CPUNET:NPB - FT	TBD
CPUNET:NPB - LU	TBD
CPUNET:NPB - MG	TBD
CPUNET:NPB - SP	TBD
IO:IOR - MPIIO Col Read	TBD
IO:IOR - MPIIO Col Write	TBD
IO:IOR - MPIIO Ind Read	TBD
IO:IOR - MPIIO Ind Write	TBD
IONET:MPI-Tile-IO - 3D Col Read	TBD
IONET:MPI-Tile-IO - 3D Col Write	TBD
Mem:HPCC - Bandwidth	TBD
Net:Graph500 - TEPS	TBD
Net:HPCC - MPI Random Access	TBD
Net:HPCC - PTRANS	TBD

Table 14: 2nd Category IV in XDMoD

Metric	Description
Avg CPU %: Idle: weighted by core-hour	TBD
Avg CPU %: System: weighted by core-hour	TBD
Avg CPU %: User: weighted by core-hour	TBD
Avg: /home write rate: Per Node weighted by node-hour	TBD
Avg: /scratch write rate: Per Node weighted by node-hour	TBD
Avg: /work write rate: Per Node weighted by node-hour	TBD
Avg: CPI: Per Core weighted by core-hour	TBD
Avg: CPLD: Per Core weighted by core-hour	TBD
Avg: CPU User CV: weighted by core-hour	TBD
Avg: CPU User Imbalance: weighted by core-hour	TBD
Avg: FLOPS: Per Core weighted by core-hour	TBD
Avg: InfiniBand rate: Per Node weighted by node-hour	TBD
Avg: Memory Bandwidth: Per Core weighted by core-hour	TBD
Avg: Memory: Per Core weighted by core-hour	TBD
Avg: Total Memory: Per Core weighted by core-hour	TBD
Avg: block sda read ops rate: Per Node weighted by node-hour	TBD
Avg: block sda read rate: Per Node weighted by node-hour	TBD
Avg: block sda write ops rate: Per Node weighted by node-hour	TBD
Avg: block sda write rate: Per Node weighted by node-hour	TBD
Avg: eth0 receive rate: Per Node weighted by node-hour	TBD
Avg: eth0 transmit rate: Per Node weighted by node-hour	TBD
Avg: ib0 receive rate: Per Node weighted by node-hour	TBD
Avg: ib0 transmit rate: Per Node weighted by node-hour	TBD
Avg: lustre receive rate: Per Node weighted by node-hour	TBD
Avg: lustre transmit rate: Per Node weighted by node-hour	TBD
Avg: mic0 receive rate: Per Node weighted by node-hour	TBD
Avg: mic0 transmit rate: Per Node weighted by node-hour	TBD
Avg: mic1 receive rate: Per Node weighted by node-hour	TBD
Avg: mic1 transmit rate: Per Node weighted by node-hour	TBD
CPU Hours: Idle: Total	TBD
CPU Hours: System: Total	TBD
CPU Hours: Total	TBD
CPU Hours: User: Total	TBD
Number of Jobs Ended	TBD
Number of Jobs Running	TBD
Number of Jobs Started	TBD
Number of Jobs Submitted	TBD
Wait Hours: Per Job	TBD
Wait Hours: Total	TBD
Wall Hours: Per Job	TBD
Wall Hours: Requested: Per Job	TBD
Wall Hours: Requested: Total	TBD

Table 15: Group By in XDMoD

Group	Description
Summary	TBD
Allocation	TBD
Field of Science	The field of science indicated on the allocation request pertain
Gateway	TBD
Grant Type	TBD
Job Size	TBD
Job Wall Time	TBD
Node Count	TBD
NSF Directorate	TBD
Parent Science	TBD
PI	TBD
PI Institution	TBD
Resource	TBD
Resource Type	TBD
Service Provider	TBD
System Username	TBD
User	TBD
User Institution	TBD
User NSF Status	TBD