The FutureGrid Testbed for Big Data

Gregor von Laszewski and Geoffrey C. Fox

Abstract In this chapter introduce you to FutureGrid, which provides a testbed to conduct research for Cloud, Grid, and High Performance Computing. Although FutureGrid has only a modest number of compute cores (about 4,500 regular cores and 14,000 GPU cores) it provides an ideal playground to test out various frameworks that may be useful for users to consider as part of their big data analysis pipelines. We focus here on the use of FutureGrid for big data related testbed research. The chapter is structured as follows. First we provide the reader with an introduction to FutureGrid hardware (Sect. 2). Next we focus on a selected number of services and tools that have been proven to be useful to conduct big data research on FutureGrid (Sect. 3). We contrast frameworks such as MPI, virtual large memory systems, Infrastructure as a Service and map/reduce frameworks. Next we present reasoning by analyzing requests to use certain technologies and identify trends within the user community to direct effort in FutureGrid (Sect. 4). The next section reports on our experience with the integration of our software and systems teams via DevOps (Sect. 5). Next we summarize Cloudmesh, which is a logical continuation of the FutureGrid architecture. It provides abilities to federate cloud services and to conduct cloudshifting; that is to assign servers on-demand to HPC and Cloud services (Sect. 6). We conclude the chapter with a brief summary (Sect. 6).

1 Introduction

FutureGrid [11, 27] is a project led by Indiana University (IU) and funded by the National Science Foundation (NSF) to develop a high performance grid test bed that will allow scientists to collaboratively develop and test innovative approaches to parallel, grid, and cloud computing. FutureGrid provides the infrastructure to researchers that allows them to perform their own computational experiments using distributed systems. The goal is to make it easier for scientists to conduct such experiments in a transparent manner. FutureGrid users will be able to deploy their own hardware and software configurations on a public/private cloud, and run

G. von Laszewski (✉) • G.C. Fox
Indiana University, Bloomington, IN, USA
e-mail: laszewski@gmail.com; gcf@indiana.edu

X. Li, J. Qiu (eds.), *Cloud Computing for Data-Intensive Applications*,
DOI 10.1007/978-1-4939-1905-5_2

their experiments. They will be able to save their configurations and execute their experiments using the provided tools. The FutureGrid test bed is composed of a high speed network connecting distributed clusters of high performance computers. FutureGrid employs virtualization technology that will allow the test bed to support a wide range of operating systems.

2 Overview of FutureGrid

2.1 Hardware Overview

FutureGrid contains a number of clusters of different types and size that are interconnected with up to a 10 GB Ethernet among its sites. The sites include Indiana University, University of Chicago, San Diego Supercomputing Center, Texas Advanced Computing Center, and University of Florida.

2.1.1 Overview of the Clusters

Table 1 provides a high level overview of the clusters currently available in FutureGrid. The biggest cluster is located at IU. It is called India and contains 128 servers with 1,024 cores. In total, we currently have 481 compute servers with 1,126 CPUs and 4,496 Cores. In addition, we have 448 GPU cores. The total RAM is about 21.5 TB. Secondary storage is about 1 PB. A more detailed table is provided in Table 2. We found that India is one of the most popular resources on FutureGrid.

Table 1 FutureGrid compute resources

Name	System type	Nodes	CPUS	Cores	TFLOPS	RAM (GB)	Storage (TB)	Site
India	IBM iDataplex	128	256	1,024	11	3,072	335	IU
Hotel	IBM iDataplex	84	168	672	7	2,016	120	UC
Sierra	IBM iDataplex	84	168	672	7	2,688	96	SDSC
Foxtrot	IBM iDataplex	32	64	256	3	768	0	UF
Alamo	Dell Poweredge	96	192	768	8	1,152	30	TACC
Xray	Cray XT5m	1	166	664	6	1,328	5.4	IU
Bravo	HP Proliant	16	32	128	1.7	3,072	128	IU
Delta	SuperMicro GPU Cluster	16	32	192		1,333	144	IU
Lima	Aeon Eclipse64	8	16	128	1.3	512	3.8	SDSC
Echo	SuperMicro ScaleMP Cluster	16	32	192	2	6,144	192	IU
		481	1, 126	[1]4696	47	22,085	1,054.2	

[1] GPU cores on machines not included

Table 2 FutureGrid cluster details

Name	Echo	Alamo	Bravo	Delta	Foxtrot	Hotel	India	Lima	Sierra	Xray
Organization	IU	TACC	IU	IU	UF	UC	IU	SDSC	SDSC	IU
Machine type	Cluster SclaeMP	Cluster	Cluster	Cluster	Cluster	Cluster	Cluster	Cluster	Cluster	Cluster
System type	SuperMicro	Dell PowerEdge M610 Blade	HP Proliant		IBM iDataPlex dx 360 M2	IBM iDataPlex dx 360 M2	IBM iDataPlex dx 360 M2	Aeon EclipseA64	IBM iDataPlex dx 340	Cray XT5m
CPU type	Xeon E5-2640	Xeon X5550	Xeon E5620	Xeon 5660	Xeon X5520	Xeon X5550	Xeon X5550	Opteron 6212	Xeon L5420	Opteron 2378
CPU speed	2.50 GHz	2.66 GHz	2.40 GHz	2.80 GHz	2.26 GHz	2.66 GHz	2.66 GHz	1.4 GHz	2.5 GHz	2.4 GHz
CPUs		192	32	32	64	168	256	16	168	168
Servers	12	96	16	16	32	84	128	8	84	1
RAM		12 GB DDR3 1333 MHz	192 GB DDR3 1333 MHz	192 GB DDR3 1,333 MHz	24 GB DDR3 1,333 MHz	24 GB DDR3 1,333 MHz	24 GB DDR3 1,333 MHz	64 GB DDR3	32 GB DDR2-667	8 GB DDR2-800
Total RAM		1,152 GB	3,072 GB	3,072 GB	768 GB	2,016 GB	3,072 GB	64 GB DDR3	2,688 GB	1,344 GB
Number of cores	144	768	128		256	672	1,024	128	672	672
Tflops		8	1.7		3	7	11		7	6
Disk size (TB)	2.8	48		15	20	120	335		72	335
Hard drives		500 GB 7.2 K RPM SAS	6x 2 TB 7.2 K RPM SATA	92 GB 7.2 K RPM SAS2	500 GB 7,200 RPM SATA	1 TB 7,200 RPM SATA	3,000 GB 7,200 RPM SATA	1 TB 7,200 RPM, 480 GB SSD	160 GB 7,200 RPM SATA Drive	6 TB Lustre
Shared storage		NFS	NFS	NFS	NFS	GPFS	NFS	ZFS	ZFS 82.2 TB	NFS
Interconnect		Mellanox 4x QDR IB	Mellanox 4x DDR IB			Mellanox 4x DDR IB	Mellanox 4x DDR IB	10 GbE Mellanox ConnectX	Mellanox 4x DDR IB	Cray SeaStar

IB = InfiniBand, Xenon = INtel Xenon, Opteron = AMD Opteron

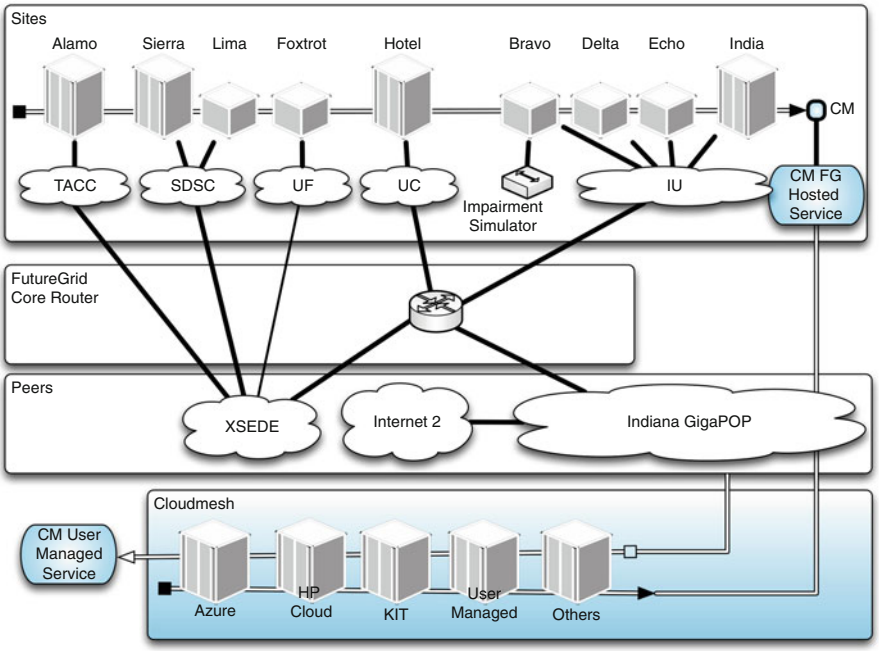


Fig. 1 High level network diagram and conceptual integration of Cloudmesh resources

2.1.2 Overview of Networking

The significant number of distinct systems within FutureGrid provides a heterogeneous distributed architecture. They are connected by high-bandwidth network links supporting distributed system research [11]. The FutureGrid network used to have a dedicated network between sites [11]. However, the network infrastructure has recently changed due to modifications related to the major network operator the National Lambda Rail. Due to these changes the operation of the network between the sites has switched from the National Lambda Rail to XSEDE and are no longer exclusive. However, this is so far no major handicap for the projects conducted on FutureGrid based on our project portfolio. The current high level network diagram is depicted in Fig. 1. Hence, the core resources to FutureGrid at SDSC, IU, TACC, and UF are now all connected via the XSEDE network and integrated via the FG core router in Chicago. Within the IU network additional clusters are integrated and are described in more detail in Sect. 2.1.1.

A Spirent H10 XGEM Network Impairment emulator [14] can be collocated with resources at Indiana University, to enable experiments to include network latency, jitter, loss, and errors to network traffic.

In addition we have added several components that are related to a special software service called Cloudmesh, which we explain in more detail in Sect. 6.

Table 3 Storage resources of FutureGrid

System type	Capacity (TB)	File system	Site
Xanadu 360	180	NFS	IU
DDN 6620	120	GPFS	UC
Sunfire x4170	96	ZFS	SDSC
Dell MD3000	30	NFS	TACC
IBM dx360 M3	24	NFS	UF

2.1.3 Overview of Storage

FutureGrid does not provide capacity for long-term storage or long-term experiments. FutureGrid has a limited amount of storage space and users are requested to remove their storage after use. However, users with special needs may be accommodated by special storage setups. The list of storage services is shown in Table 3.

3 Services and Tools for Big Data

FutureGrid offers a very rich environment to its users. We can categorize them in a stacked service architecture as depicted in Fig. 2. We distinguish the following categories: Cloud PaaS, IaaS, GridaaS, HPCaaS, TestbedaaS, which we will explain in more detail in the next sections. The services in these categories are integrated in our general FutureGrid high-level architecture depicted in Fig. 3. More details about the architecture can be found in [11, 27]. Within this paper we will focus on describing services that have been explicitly used for big data research in FutureGrid.

3.1 Testbed as a Service (TestbedaaS)

It is a well-accepted paradigm that today a lot of research is carried out by interdisciplinary scientific teams. Thus, FutureGrid provides an advanced framework to manage user and project affiliation and propagates this information to a variety of subsystems constituting the FG service infrastructure. This includes operational services (not explicitly mentioned in Fig. 2) to deal with authentication, authorization and accounting. In particular, we have developed a unique metric framework that allows us to create usage reports from our entire Infrastructure as a Service (IaaS) frameworks. Repeatable experiments can be created with a number of tools including Pegasus, Precip and Cloudmesh. VMs can be managed on high

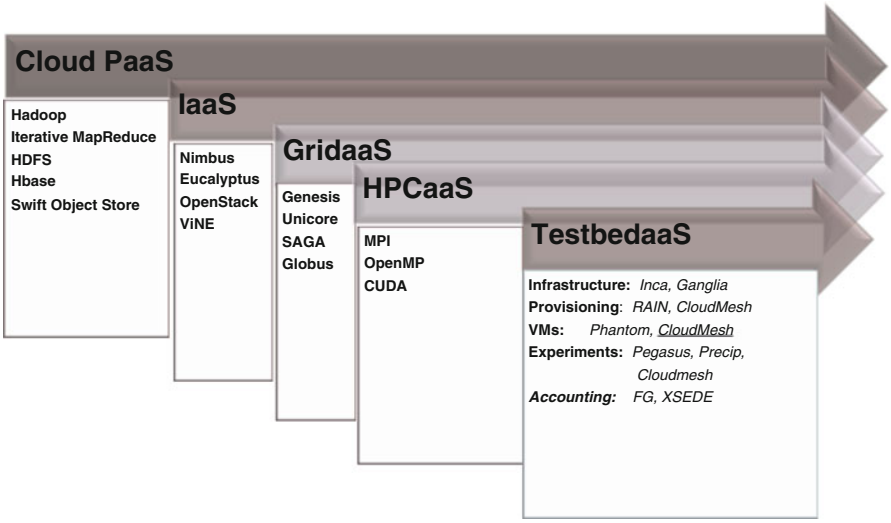


Fig. 2 FutureGrid high-level user services

level either via Cloudmesh (see Sect. 6). Provisioning of services and images can be conducted by RAIN [9, 10]. Infrastructure monitoring is enabled via Nagios [7], Ganglia [17], and Inca [22] and our own cloud metric system [29].

3.2 Traditional High Performance Computing as a Service (HPCaaS)

Within the traditional High Performance Computing (HPC) services FG offers a traditional MPI/batch queuing system and a virtual large memory system that are beneficial for big data calculations.

3.2.1 MPI and Batch Queues

The traditional HPC environment provided by queuing systems and Message Passing Interface (MPI) programs creates a suitable infrastructure not only for simulations, but also for the analysis of large data. However, considerable amount of work has to be conducted to optimize the available infrastructure to deal with distributed domain decompositions. Optimized use via traditional HPC has been successfully demonstrated for many biological applications. Additionally, the existence of a queuing system can provide some advantages when the available resources are over utilized while sharing the resources with other users. This has

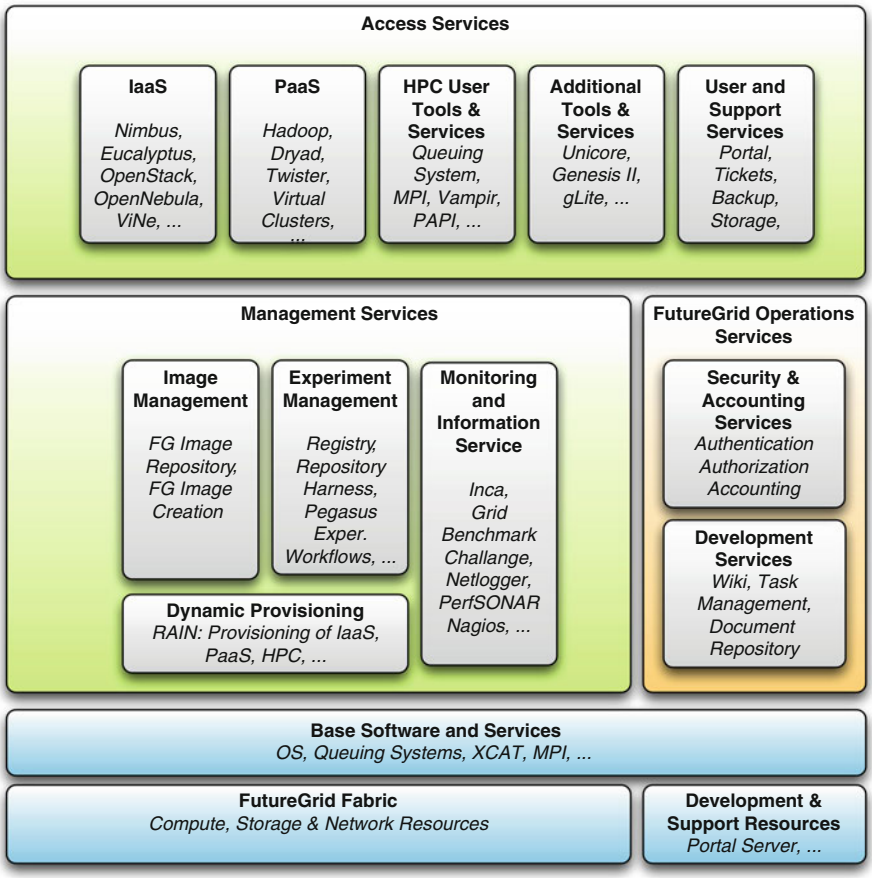


Fig. 3 FutureGrid high-level architecture

been especially useful in FutureGrid to support educational activities for classes with many users that, for example, want to test map reduce activities controlled by a queuing system as described in Sect. 3.5.1.

3.2.2 Virtual Large-Memory System

One of the demands often posed in big data analysis it to place the data as much as possible into memory to speed up calculations and in some cases to fit the entire dataset into memory. However, this analysis may come at a cost as, for example, the use of HPC computing via MPI adds additional programming complexity within a cluster. Therefore, it is desirable to virtualize the memory from multiple servers in a cluster to provide one big memory system that can be easily accessed by the

underlying software. One such implementation, vSMP by ScaleMP [11, 21]. vSMP is installed on the FutureGrid echo cluster that has 16 servers and can access up to 3 TB in shared virtual memory.

3.3 Grid as a Service (GridaaS)

Not surprisingly the demand for computational Grids on FutureGrid has been relatively small. While we saw few requests for Globus we decided to focus on the installation of more popular systems. The low use can be explained by the availability of large Grid production infrastructure elsewhere such as in XSEDE and based on the move of the community away from complex Grid solutions to either cloud computing or even back to more traditional batch processing solutions. Furthermore, toolkits such as the CoG Kit also known as jglobus [24, 26, 28] have provided enough abstractions for users that experimenting with such technologies has become less prominent and can be made on the client side while interfacing to production Grid services instead of testbeds.

3.4 Infrastructure as a Service (IaaS)

One of the main features of FutureGrid is to offer its users a variety of infrastructure as a service frameworks [25, 31]. These frameworks provide virtualized resources to the users on top of existing cyberinfrastructure fabric. This includes but is not limited to virtualized servers, storage, network, disk, and other IaaS related services. In FutureGrid the most common hypervisor that runs the virtual machines as guest on the underlying operating system is KVM. Some resources also run XEN, however most recently the demand for KVM has increased and some services will be switched from XEN to KVM. Through the ability to provide large numbers of virtual machines to the users, the access mode to utilize resources, in contrast to traditional HPC, has been changed from a reservation-based service to an on-demand service. This comes with the benefit that if enough resources are available they will be immediately allocated to the user. However, if not enough resources can be offered, the system will define the request and return with an error. Based on our experience with FutureGrid over the last couple of years, it is advantageous to offer a mixed operation model. This includes a standard production cloud that operates on-demand, but also a set of cloud instances that can be reserved for a particular project. We have conducted this for several projects in FutureGrid, including those that required dedicated access to resources as part of big data research such as classes [18, 19] or research projects with extremely large virtual machines [20].

The IaaS services that are offered in FutureGrid contain the following:

OpenStack has become most recently, next to HPC, the most requested service in FutureGrid based on newly started projects. OpenStack is an open source

cloud infrastructure as a service framework to deliver public and private clouds. It contains a number of components that together build a powerful and flexible set to create a cloud service offering. Services include a compute service, and object storage, an image service, a monitoring service, and an orchestration service. OpenStack has received considerable momentum due to its openness and the support of companies. Within FutureGrid OpenStack clouds are currently deployed on India, Sierra, Hotel, and Alamo, while currently India provides the most up to date services.

Nimbus is an open source service package allowing users to run virtual machines on FutureGrid hardware. Just as in OpenStack users can upload their own virtual machine images or customize existing ones. Nimbus, next to Eucalyptus is one of the earlier frameworks that make managing virtual machines possible. Nimbus provides a basic set of cloud services including services to orchestrate the deployment of virtual machines. However, Eucalyptus and OpenStack now also provide such services.

Nimbus provides a selected subset of AWS protocols such as EC2. Accounting of Nimbus VMs does not currently provide features for project management. Such group-based and role based user management is essential for proper administrative resource and project management and is provided by other IaaS frameworks. In Nimbus it is only conducted on a user-by-user basis. This has significant implications on user management as in large-scale deployments project management features are highly desired but are not offered in Nimbus. Although, single users may not be interested in this feature, it is essential to provide proper project management of groups of users.

Eucalyptus is an open source software IaaS framework for cloud computing. Eucalyptus provides an Amazon Web Services (AWS) compliant EC2-based web service interface to its users enabling the easy integration between a local cloud managed with Eucalyptus and AWS. However, as other IaaS frameworks such as OpenStack also provide EC2 interfaces for many application users OpenStack has become a viable alternative.

Which of the IaaS frameworks to choose is a question that is not that easy to answer. Many of our projects evaluate several of them in order to choose the one best suited for their use case. At other times users chose a framework that they had previously successfully used. Over time the quality of the IaaS framework has significantly changed. Within the last year OpenStack has become the most popular platform on FutureGrid.

3.5 Cloud Platform as a Service (PaaS)

3.5.1 Map Reduce

Map reduce models have been familiar to the programming and distributed computing community for a long time and have been historically associated with the

functional programming's map and reduce. However the map and reduce framework introduced recently [8] distinguishes itself from such efforts while applying it repeatedly, with fault tolerance on a very large distributed data set [4].

Instead of bringing the data to the computer in map reduce application we often use the concept of bringing the computing to the data. This makes a lot of sense when we assume that a large number of data is distributed over many servers and repeated search queries are cast to find results across them (as in the case of Google motivating map/reduce).

In general, we can define a *map* step that takes the input problem and divides it into smaller sub-problems distributing it among worker nodes. The map function is then executed on the data distributed on the various servers. The *reduce* step collects the answers of the subproblem and combines them in some fashion (Fig. 4).

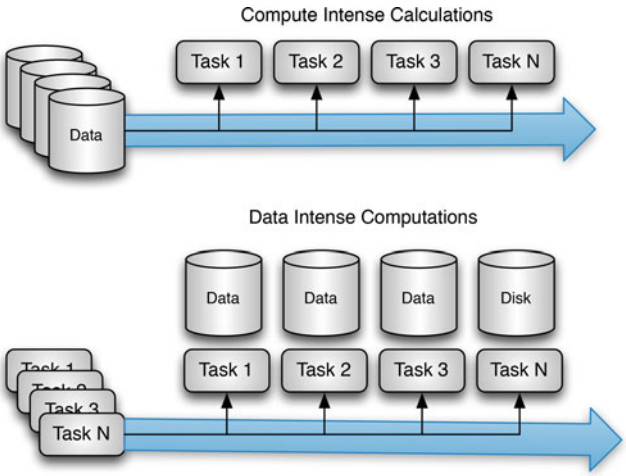


Fig. 4 Bring the data to the computation vs. bring the computation to the data

Hadoop

Hadoop [1] is an Apache project delivering an opensource software that uses the map/reduce framework in a distributed environment while focusing on scalability and reliability. Its design includes the Hadoop File System (HDFS) which provides an easy-to-use file system to distribute the data among the servers on which the calculations will be executed. Hadoop is designed to deal with faults through redundancy which is an important feature when conducting data analysis on very large distributed databases [1]. Hadoop is written in Java and provides the essential map reduce functionality and allows the system to be configured for existing hardware.

myHadoop

MyHadoop [15, 16], which is installed on many of the compute clusters in FutureGrid, enables users to launch Hadoop clusters via traditional high-performance compute clusters. For this, it utilizes the underlying batch scheduling system.

The reasons for managing Hadoop jobs via a batch system are manifold. First, the available infrastructure is resource constrained, and utilization of disks and compute resources must be specially accounted for to allow shared usage by many users. This naturally happens in the educational research community quite frequently. Second, to efficiently utilize the compute and data infrastructure researchers may not run Hadoop or MPI jobs continuously. At times they may need a Hadoop environment. At other times they may prefer a traditional message passing environment while at the same time being under resource constraints.

The idea of myHadoop is to submit a job to the queuing system that sets up a Hadoop cluster for the length of the reservation and the researcher can then use it to conduct experiments either via predefined jobs or in interactive mode. This is achieved by first identifying a number of resources via the scheduler, followed by the deployment of the Hadoop software across the identified servers. The user will then be presented with information on how to access this newly provisioned Hadoop cluster. MyHadoop, in its new version [16] is supported for Oracle Grid Engine (formerly known as Sun Grid Engine), PBS, and SLURM.

Once Hadoop has been initialized, it can be accessed through regular job scripts as shown in Fig. 5. This example script uses eight nodes. It is important to set the processor per node to 1 to assure the various Hadoop daemons are scheduled on different servers. The rest of the script is not depicted as it contains the actual details on setting up Hadoop via the script and is beyond the scope of this chapter. The user should replace the text in $< \ldots >$ to customize the job. As Hadoop is a user level program, it is also possible to run a usermodified version of Hadoop which helps in adding new features or trying out newer versions of Hadoop than the default version that is installed for my Hadoop. The FutureGrid manual provides more details on how to practically use myHadoop on FutureGrid [13].

```
#!/bin/bash
#PBS -q <queue_name>
#PBS -N <job_name>
#PBS -l nodes=8:ppn=1
#PBS -o <output file>
#PBS -e <error_file>
#PBS -A <allocation>
#PBS -V
#PBS -M <user email>
#PBS -m abe

...further details omitted
```

Fig. 5 PBS script to start hadoop

Twister

Twister [6] is an extension to MapReduce to allow more easily the introduction of iterative map reduce processes. In addition twister has introduced a number of concepts including distinction between static and variable data, long running tasks, publish/subscriber based communication, and various other enhancements. Twister is developed at Indiana University, and is used as part of classes on distributed systems and other educational activities; hence, it reaches popularity within FutureGrid.

Virtual Clusters

In addition to the map/reduce platforms offered on FutureGrid, it is also possible to deploy virtual clusters. One of the earliest such frameworks has been showcased by von Laszewski [30] while deploying a SLURM cluster. Such a cluster can then be used as a teaching tool or provides additional mechanisms to custom create queues and reservations. However, the most advanced feature of FutureGrid will be via Cloudmesh, which will allow the deployment of clusters not only in virtual machines, but on baremetal.

4 FutureGrid Usage

When offering services such as FutureGrid to the community, we have to analyze and predict which services may be useful for the users. We have therefore established a number of activities that monitor external and internal data. Externally, we look, for example, at information provided by Gartners technology hype curve [2] or Google trend information as shown in Fig. 6. From Google Trend data we observe that the popularity of Grid computing has been very low in the recent years and much attention has shifted to cloud computing. Therefore we removed this information from the figure and focus exemplary on cloud related terms such as *Cloud Computing*, *Big Data*, *OpenStack*, and *VMWare*. From this information we see that all but VMWare are rising, with Cloud Computing dominating the Google trends in comparison to the others. This trend is important as it shows a shift in the cloud computing community buzz away from a traditional commercial market leader in virtualization technology. We believe that is correlated with a large number of vendors offering alternative products and services while at the same time the novelty from VMWare is reduced.

To give an informal overview of the more than 300 projects conducted on FutureGrid, we have taken their titles and displayed them in a word cloud (see Fig. 7). Additionally, we have taken keywords that are provided by the project leads and also displayed them in a word cloud (see Fig. 8). Although the images do not give quantitative perspective about the project it helps to identify some rough idea

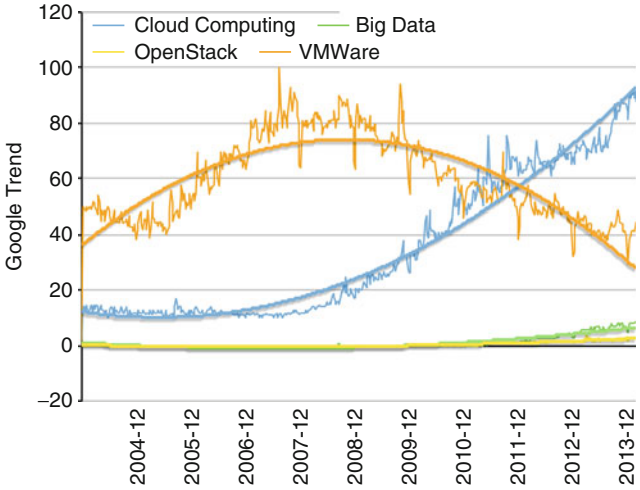


Fig. 6 Google trends

about the activities that are ongoing in FutureGrid. As expected the terms cloud computing and terms such as mapreduce, OpenStack, Nimbus, and Eucalyptus appear quite frequently. It is, therefore, worthwhile to analyze this data in a more quantitative form.

FutureGrid supports a rich set of projects, of which many are principally related directly or indirectly to big data systems. In Table 4 we list the areas of research and the number of projects conducted in these areas over a time period identified between November 2011 and December 2013. One of the focal observations is that the majority of projects are related to computer science research which can be found in the table in more detail. Domain Science and Education related projects take on a large portion.

As part of our project management in FutureGrid, we have designed a simple project application procedure that includes prior to a project being granted access, gathers information about which technologies are anticipated to be used within the project. The list of technologies is fairly extensive and includes Grid, HPC, and Cloud computing systems, services, and software. However, for this paper we will focus primarily on technologies that are dominantly requested and depicted in Fig. 9. Clearly we can identify the trend that shows the increased popularity of OpenStack within the services offered on FutureGrid. Nimbus and Eucalyptus are on a significant downward trend. ObenNebula was also at one point more requested than either Nimbus or Eucalyptus, but due to limited manpower an official version of OpenNebula was not made available on FutureGrid. As we have not offered it and pointed it out on our Web page, requests for OpenNebula have vanished. However, we have internally used OpenNebula for projects such as our Cloudmesh rain framework. All other sixteen technologies are relatively equally distributed over

Fig. 7 Project title word cloud

Fig. 8 Project keyword word cloud

the monitoring period. The lesson that we took form this is that FutureGrid has put recently more emphasis in offering OpenStack services.

From the overall project information we have also analyzed the frequency of the number of project members within the project and show it in Fig. 10. Here we depict on the abscissa, classes of projects with varying members. Assume we look at the abscissa at the value of 10. This means that these are all projects that have project

Table 4 FutureGrid supports a rich set of projects, of which many are importantly related directly or indirectly to big data systems

Discipline	Count
Domain science	44
Education[a]	42
Technology evaluation	19
Core virtualization	17
Programming models	12
Cyberinfrastructure	11
Security and privacy	10
Data systems	10
Resource management	9
Distributed clouds and systems	8
Artificial intelligence	7
Cyber-physical CPS and mobile systems	5
Fault-tolerance	5
Data analytics/machine learning	5
Networking	3
Network/web science	3
Interoperability	3
Storage	2
Streaming data	2
P2P	2
Software engineering	2

[a] 90 % of which on computer science

members between 10 and its previous category, in this case 5. Hence, it will be all projects greater than 5 and smaller or equal to 10. With this classification we see that the dominant unique number of members within all projects is either one, two or three members. Then we have another class between four and ten members, and the rest with more than ten members. One of the projects had 186 registered members overall for an education class as part of a summer school. Looking at the distribution of the members and associating them with research and education projects, we find all projects with larger numbers of projects to be education projects.

When we look in more detail into the map/reduce related technology requests over the entire period FutureGrid has been active, we identified the distributions as depicted in Fig. 13. We can see that the requests for IaaS together with map/reduce technology requests dominate. HPC requests are much fewer. The reason why the other category in Fig. 13 is that high is because we have a significant number of other choices, each with a very low total count. Also, we would like to remind the reader that users can chose multiple categories for their projects. Within the category of map/reduce, users had the choice of Hadoop, Map/Reduce, or Twister as a technology. The breakdown of these choices is shown in the right part of Fig. 13 dominated by the choice for Map/Reduce and Hadoop representing 85 % of all choices.

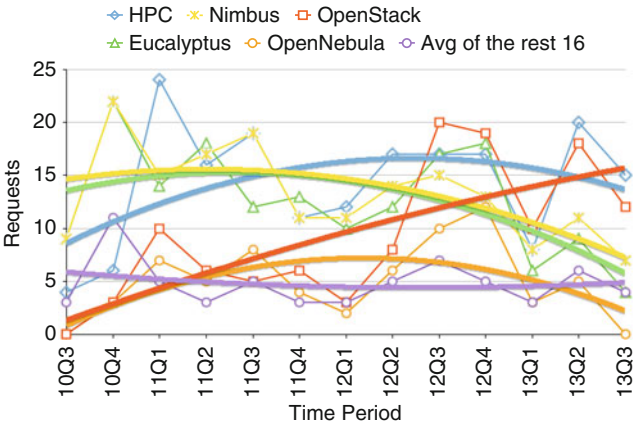


Fig. 9 Requested technologies by project

Next we have analyzed all projects that requested either mapreduce, hadoop, twister, MPI and ScaleMP (147 of all 374 active projects, which is 39 % of all projects) and categorized them by discipline as shown in Fig. 11. In contrast to XSEDE, which provides a production HPC system to the scientific community, the usage of FutureGrid for map reduce frameworks is dominated with 50 % by computer science related projects followed by education with 19 %.

Looking further into this data, we present in Fig. 12 the number of projects in a particular category as well as the Fraction of technologies within a discipline. As we are focusing in this paper on the impact on big data, we have looked in particular at requests for mapreduce, Hadoop, and twister, while also looking at requests for MPI and ScaleMP. It is interesting to note that the perceptual distribution of the technologies among these projects is about constant, if we exclude technology evaluations and interoperability. As MPI is more popular with domain sciences, we find a slight increase in projects requesting MPI (Fig. 13). However, with the life sciences we see the opposite as map/reduce and associated technologies are more popular here. MPI and ScaleMP are not much requested as part of technology evaluations and interoperability experimentation as they either project a very stable framework and do not require evaluation, or the question of interoperability is not of concern for most of the projects.

5 System Management

The goal of FutureGrid is to offer a variety of services as part of its testbed features, thereby going beyond services that are normally offered by data and supercomputing centers for research. This provides a number of challenges that need to be overcome

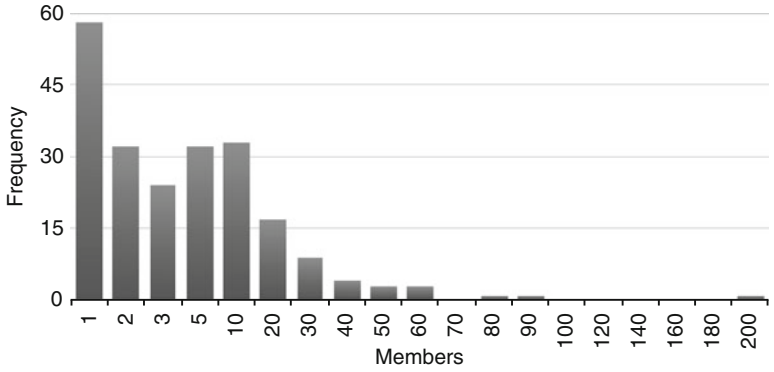


Fig. 10 Project frequency

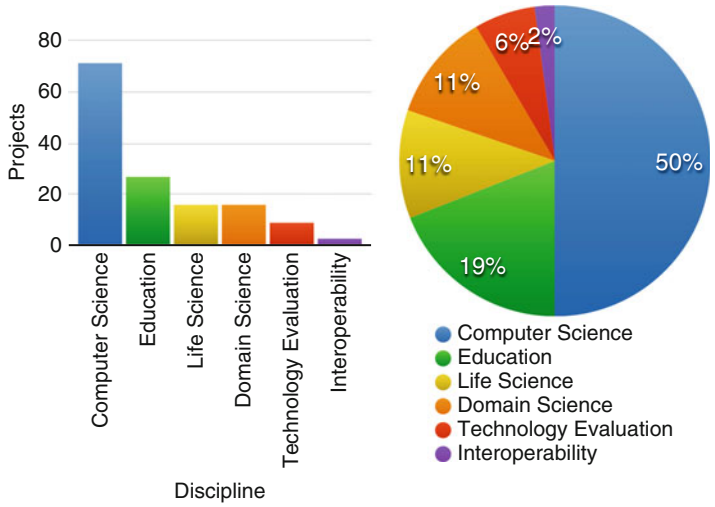


Fig. 11 Distributon of project disciplines

in order to efficiently manage the system and provide services that have never been offered to users as they exist on FutureGrid.

5.1 Integration of Systems and Development Team

FutureGrid started initially with a model where the systems team and the software team were separated. An unnecessary wall between teams was erected that resulted in multiple challenges:

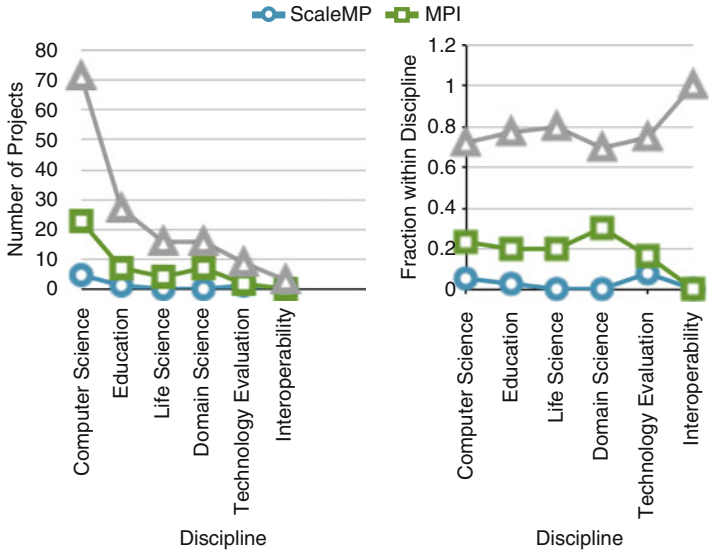


Fig. 12 Requests by of technologies by discipline within a project. (*Triangle*) Map reduce, Hadoop, or Twister, (*square*) MPI, (*circle*) ScaleMP

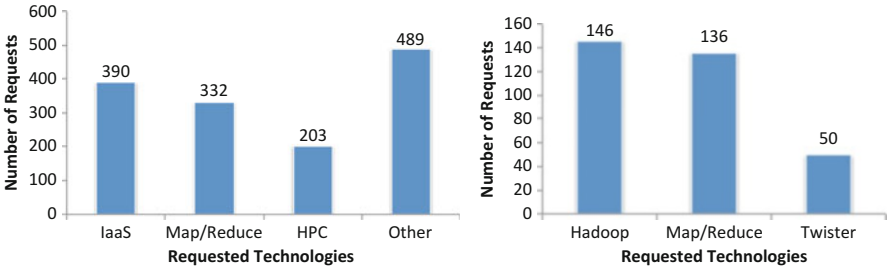


Fig. 13 Map reduce related technology requests

1. The system setup and management were completely separated from the software development team focusing mostly on the deployment of existing technologies. However the technologies deployed were themselves under heavy development and required intercorrelations between developers and system teams.
2. The deployed system was complex, but its deployment was documented to a limited extent, which resulted in developers having insufficient information to utilize the system properly or to know what had been deployed.
3. Lack of trust by the systems team did not allow the software team to have a valid development environment as proper privileges were not issued to the developers. As the development team needed to use privileged system services, the development could not be carried out.

4. The software developed needed a testbed within the testbed that was not necessarily reflecting the actual system setup.

Together, these issues made it extremely difficult, if not impossible, to further any development in regards to the design of a testbed infrastructure as proposed by our original ambitious goals.

To overcome these difficulties it was decided early on in the project that the systems team must be integrated in some fashion into the software team and become part of the development process. This integration is not an isolated instance within FutureGrid, but is also executed in many modern data centers and is now recognized with its own term called *DevOps*.

5.2 DevOps

DevOps is not just a buzzword from industry and research communities. It provides value added processes to the deployment and development cycles that are part of modern data centers. It can today be understood as a software development method that stresses collaboration and integration between software developers and information technology professionals such as system administrators.

While using an infrastructure such as clouds we recognized early on that the lifetime of a particular IaaS framework is about 3–6 months before a new version is installed. This is a significant difference to a traditional High Performance Computing Center that is comprised of many software tools experiencing much longer life spans. This is not only based on security patches but significant changes, for example, in the evolving security infrastructure and user services, as well as, the deployment of new services that become available in rapid procession.

This rapid change of the complex infrastructure requires rethinking how systems in general are managed and how they can be made available to the development teams. While previously it may have been enough to install updates on the machines, DevOps frameworks provide the developer and system administrators a way to create and share environments that are used in production and development while at the same time increasing quality assurance by leveraging each other's experiences (see Fig. 14).

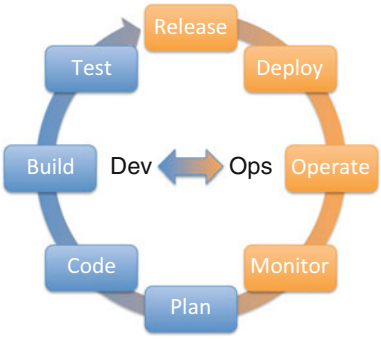
5.2.1 DevOps Cycle

While combining the steps executed by the development and operational team from planning to coding, building and testing, to the release, deployment and operation and monitoring (see Fig. 15), each of the phases provides a direct feedback between the DevOps team members, thus shortening the entire development phase. It also allows testing out new services and technologies in a rapid progression. Hence, it

is possible to roll out new developments into production much faster. This leads to a much more rapidly integrated cycle than would not be possible without the correlation between development and operation.

Fig. 14 DevOps intersection

Fig. 15 DevOps cycle



5.2.2 DevOps Supporting Tools

A number of tools are available that make the introduction of DevOps strategies more efficient. The first is the need for an simplified communication pathway to manage tasks not only between developers but also between users. Thus the ideal system would provide a complete integration of a project management system that allows managing tasks for both developers and operators, but also to easily integrate tickets and transform them into tasks. In XSEDE and other supercomputing centers a system called RT [5] is typically used for user ticket management. Other systems such as jira, mantis, and basecamp are often used to manage the software and systems related tasks. Unfortunately, personal or organizational constraints often prevent the integration of the two systems and additional overhead is needed to move user tickets into tasks and the development cycle. Within FutureGrid, as part of our opensource development, we experimented extensively with jira as systems

and ticketing system [3] revealing that newest development in such areas motivated by DevOps teams led to tools that support the overall cycle including user ticket management in a single system (see Fig. 16). However, the integration of FutureGrid within the overall much larger XSEDE effort made it not possible to switch from RT to jira for user ticket management. To stress this user integration we term this framework *UseDevOps*. Tools to integrate Development and Operation deployment include puppet, chef, ansible, cfengine and bcfg2. While FutureGrid started out with bcfg2 we have since switched to other tools due to their prevalence within the community. Chef, puppet, and ansible have significant amount of traction. Due to expertise within our group we currently explore chef and ansible.

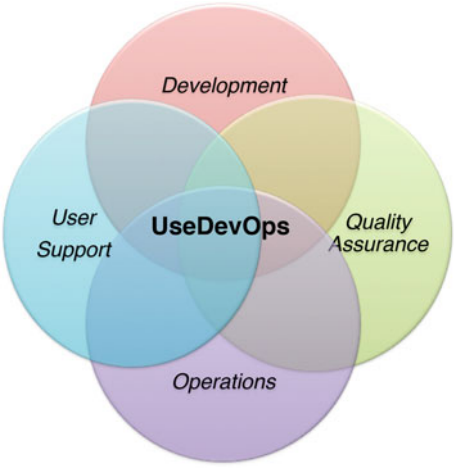


Fig. 16 User Support integrated into DevOps leads to UseDevOps

5.3 Support for Education

To support the many educational and research projects on FutureGrid, we have provided, through a portal, a significant amount of material on how to use the discussed services. In addition, we realized that not every educational project has users with advanced computer experience, therefore we provide for such projects a streamlined user interface rather than having the users fight with complex command line syntax and parameters. For example, we provided for a recent MOOC on big data taught with resources on FutureGrid the basic functionality not only to start VMs as part of the IaaS framework, but also to deploy sophisticated images that contain preinstalled software and allow services to be hosted by the users such as iPython, R and much more. This was implemented on top of OpenStack while utilizing the newest OpenStack services such as Heat. The management of the VMs and starting of the iPython server was controlled by a python application

that provides the user with a menu system. Thus, the management of them became literally as easy as pressing 1, 2, 3, ... in the menu. For other classes, we have also provided completely separate OpenStack deployments as the teachers were afraid that students would not have enough resources due to the shared environment. However, we learned from this that the teachers overestimated the actual utilization of the project and many resources were not used. Based on this analysis we now have a model to justify the creation of more relaxed access policies and can justify that even classes should be utilizing the public region that are provided by FutureGrid. If resource contention would become an issue, we could set aside a special region for a limited amount of time. Reconfiguration needs have also arisen where one day a class may want to explore traditional MPI, while the next they want to experiment with Hadoop. Furthermore, we identified that several users wanted to combine various cloud IaaS platforms in order to avoid resource over-provisioning or were interested in combining all resources.

6 Cloudmesh

At [12] we find an extensive set of information about Cloudmesh that is cited within this section.

From the experience with FutureGrid we identified the need for a more tightly integrated software infrastructure addressing the need to deliver a software-defined system encompassing virtualized and baremetal infrastructure, networks, application, systems and platform software with a unifying goal of providing Cloud Testbeds as a Service (CTaaS). This system is termed Cloudmesh to symbolize

(a) The creation of a tightly integrated mesh of services targeting multiple IaaS frameworks.
(b) The ability to federate a number of resources from academia and industry. This includes existing FutureGrid infrastructure, Amazon Web Services, Azure, HP Cloud, Karlsruhe, using not only one IaaS framework, but many.
(c) The creation of an environment in which it becomes more easy to experiment with platforms and software services while assisting to deploy them effortlessly.

In addition to virtual resources, FutureGrid exposes baremetal provisioning to users, but also a subset of HPC monitoring infrastructure tools. Services will be available through command line, API, and Web interfaces.

6.1 Functionality

Due to its integrated services Cloudmesh provides the ability to be an onramp for other clouds. It also provides information services to various system level sensors to give access to sensor and utilization data. They internally can be used to optimize

the system usage. The provisioning experience from FutureGrid has taught us that we need to provide the creation of new clouds, the repartitioning of resources between services (cloud shifting), and the integration of external cloud resources in case of over provisioning (cloud bursting). As we deal with many IaaS we need an abstraction layer on top of the IaaS framework. Experiment management is conducted with workflows controlled in shells [23], Python/iPython, as well as systems such as OpenStack's Heat, Accounting is supported through additional services such as user management and charge rate management. Not all features are yet implemented. Figure 17 shows the main functionality that we target at this time to implement.

6.2 Architecture

The three layers of the Cloudmesh architecture include a Cloudmesh Management Framework for monitoring and operations, user and project management, experiment planning and deployment of services needed by an experiment, provisioning and execution environments to be deployed on resources to (or interfaced with) enable experiment management, and resources (Fig. 18).

6.2.1 System Monitoring and Operations

The management framework contains services to facilitate FutureGrid day-to-day operation, including federated or selective monitoring of the infrastructure. Cloudmesh leverages FutureGrid for the operational services and allows administrators to view ongoing system status and experiments, as well as interact with users through ticket systems and messaging queues to inform subscribed users on the status of the system.

The Cloudmesh management framework offers services that simplify integration of resources in the FutureGrid nucleus or through federation. This includes, for user management, access to predefined setup templates for services in enabling resource and service provisioning as well as experiment execution. To integrate IaaS frameworks Cloudmesh offers two distinct services:

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

For (b) several toolkits exist to create user-based federations, including our own abstraction level which supports interoperability via libcloud, but more importantly it supports directly the native OpenStack protocol and overcomes limitations of the EC2 protocol and the libcloud compatibility layer. Plugins that we currently develop will enable access to clouds via firewall penetration, abstraction layers for clouds with few public IP addresses and integration with new services such

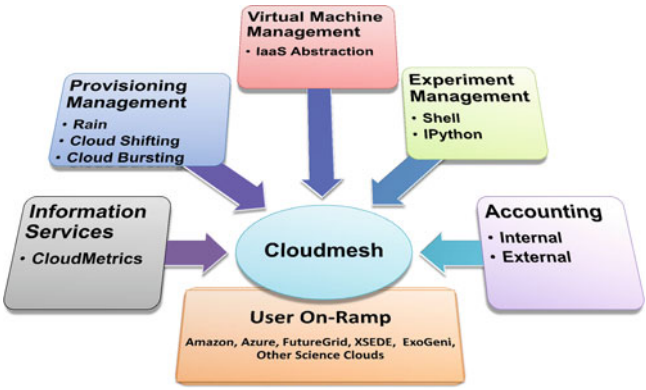


Fig. 17 CM functionality

as OpenStack Heat. We successfully federated resources from Azure, AWS, the HP cloud, Karlsruhe Institute of Technology Cloud, and four FutureGrid clouds using various versions of OpenStack and Eucalyptus. The same will be done for OpenCirrus resources at GT and CMU through firewalls or proxy servers.

Additional management flexibility will be introduced through automatic cloud bursting and shifting services. While cloud bursting will locate empty resources in other clouds, cloud shifting will identify unused services and resources, shut them down and provision them with services that are requested by the users. We have demonstrated this concept in 2012 moving resources for more than 100 users to services that were needed based on class schedules. A reservation system will be used to allow for reserved creation of such environments, along with improvements of automation of cloud shifting.

6.2.2 User and Project Services

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

6.2.3 Accounting and App Store

To lower the barrier of entry, Cloudmesh will be providing a shopping cart which will allow checking out of predefined repeatable experiment templates. A cost is

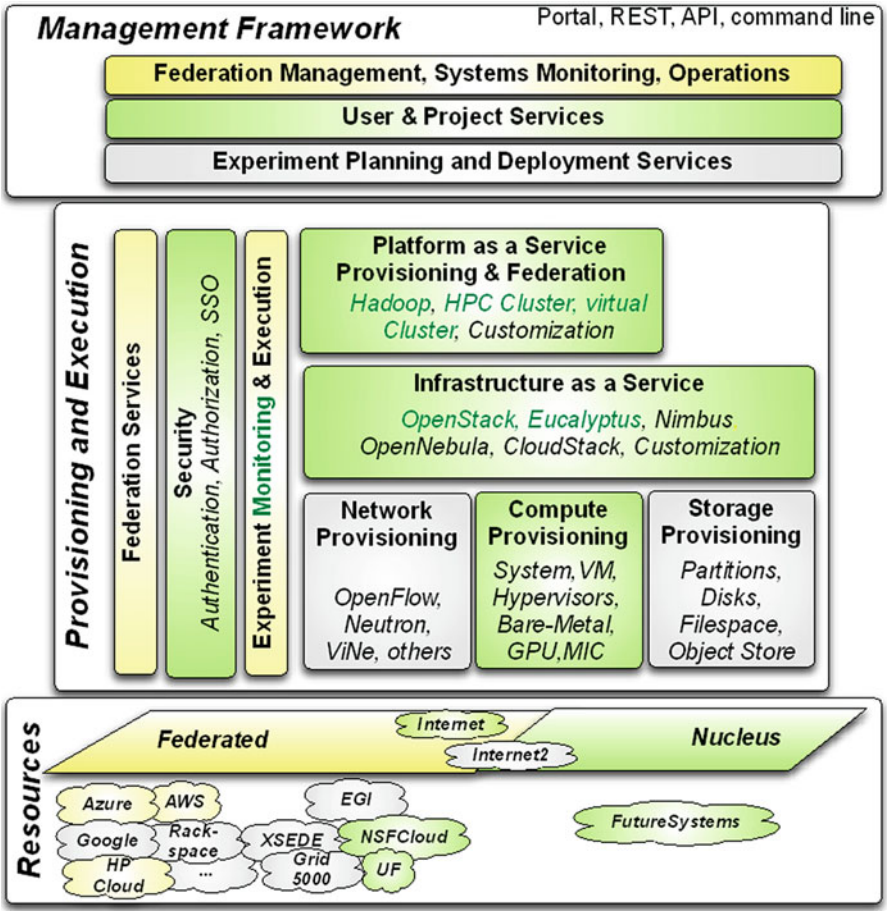


Fig. 18 CM architecture

associated with an experiment making it possible to engage in careful planning and to save time by reusing previous experiments. Additionally, the Cloudmesh App Store may function as a clearing-house of images, image templates, services offered and provisioning templates. Users may package complex deployment descriptions in an easy parameter/form-based interface and other users may be able to replicate the specified setup.

Due to our advanced Cloudmesh Metrics framework, we are in the position to further develop an integrated accounting framework allowing a usage cost model for users and management to identify the real impact of an experiment on resources. This will be useful to avoid over-provisioning and inefficient resource usage. The cost model will be based not only on number of core hours used, but also the capabilities of the resource, the time, and special support it takes to set up the experiment. We will expand upon the metrics framework of FutureGrid that allows

measuring of VM and HPC usage and associate this with cost models. Benchmarks will be used to normalize the charge models.

6.2.4 Networking

We have a broad vision of resource integration in FutureGrid offering different levels of control from baremetal to virtual machine and platform management. We also offer the ability to utilize resources in a distributed environment. Likewise, we must utilize networks offering various levels of control, from standard IP connectivity to completely configurable SDNs, as novel cloud architectures will almost certainly leverage NaaS and SDN alongside system software and middleware. FutureGrid resources will make use of SDN using OpenFlow whenever possible, however, the same level of networking control will not be available in every location.

6.2.5 Monitoring

To serve the purposes of CISE researchers, Cloudmesh must be able to access empirical data about the properties and performance of the underlying infrastructure beyond what is available from commercial cloud environments. To accommodate this requirement, we have developed a uniform access interface to virtual machine monitoring information available for OpenStack, Eucalyptus, and Nimbus. In the future, we will be enhancing the access to historical user information. Right now they are exposed through predefined reports that we create on a regular basis. To achieve this we will also leverage the ongoing work while using the AMPQ protocol. Furthermore, Cloudmesh will provide access to common monitoring infrastructure as provided by Ganglia, Nagios, Inca, perfSonar, PAPI and others.

6.3 Cloud Shifting

We have already demonstrated via the RAIN tool in Cloudmesh that it is possible to easily shift resources between services. We are currently expanding upon this idea and developing more easy to use user interfaces that assist administrators and users through role and project based authentication to move resources from one service to another (see Fig. 19).

6.4 Graphical User Interface

Despite the fact that Cloudmesh was originally a quite sophisticated command shell and command line tool, we have recently spent more time in exposing this

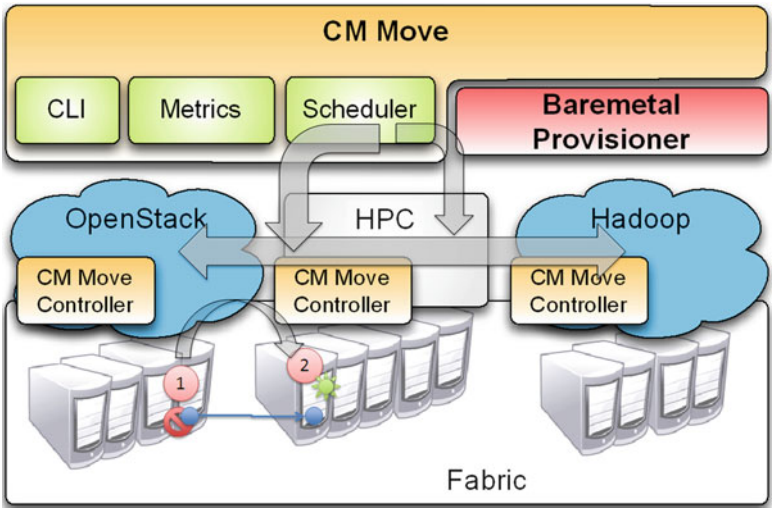


Fig. 19 Shifting resources makes it possible to offer flexibility in the service distribution in case of over or underprovisioning

functionality through a convenient Web interface (Fig. 20). Some more popular views in this interface are depicted in Fig. 21 hinting at how easy it is with a single button to create multiple VMs across a variety of IaaS. This not only includes resources at IU but also at external locations, making it more practical for users.

Hence, this easy management provides a more sophisticated experience for the user while associating one-click deployments. These deployments include the ability to instantiate virtual clusters, Hadoop environments, and other more elaborate setups. We provide an early prototype screenshot in Fig. 22.

6.5 Command Shell and Command Line Interface

Cloudmesh contains the ability to access much of its functionality through a commandline interface. This is enabled through a command shell that can function both as regular Linux command as well as command shell. The command shell can be invoked with `cm` on the regular Linux shell, or by specifying a number of parameters to call it without starting the interactive shell. The commands accessible to the users allow the management of virtual machines, and bare metal provisioning. To find out the detailed option of each command one can invoke a `help` command. An important property of this shell is that it can be easily extended and plugins can be distributed not only in a system wide plugin directory, but also in one provided by the user. This makes it possible to customize the shell commands available

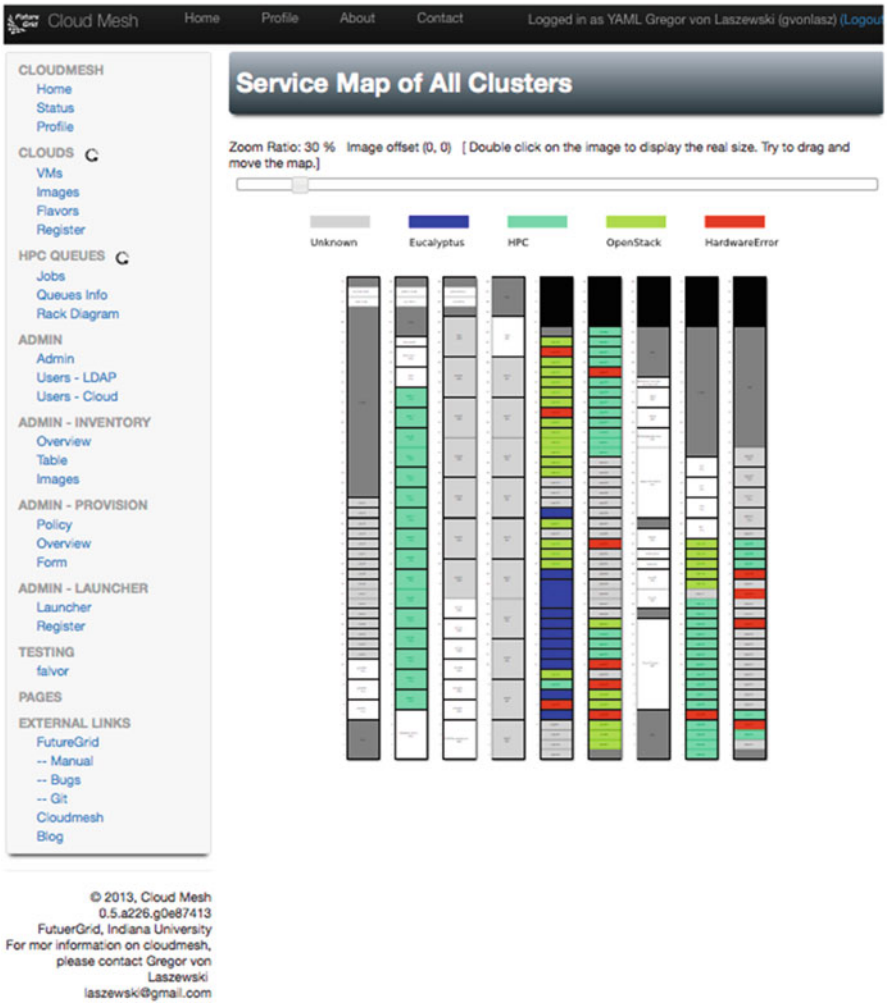


Fig. 20 Monitoring the Service distribution of FutureGrid with Cloudmesh

based on user needs by either removing unnecessary commands or introducing new ones. For example it would be possibly for users only interested in virtual machine management to remove the commands related to baremetal provisioning. Figure 23 shows the startup of the command shell and the invocation of the help command to list the available commands for this user. Please note that all commands are yet implemented. We have categorized some of the commands as cloud commands listed in a special section in the help. Figure 24 shows the use of the first command in that list called `cloud` that lists the available clouds for the user. Important to note is that Cloudmesh can be set up as a server so that the GUI and the command shell share variables that are managed in a jointly used database. Thus, it is possible to

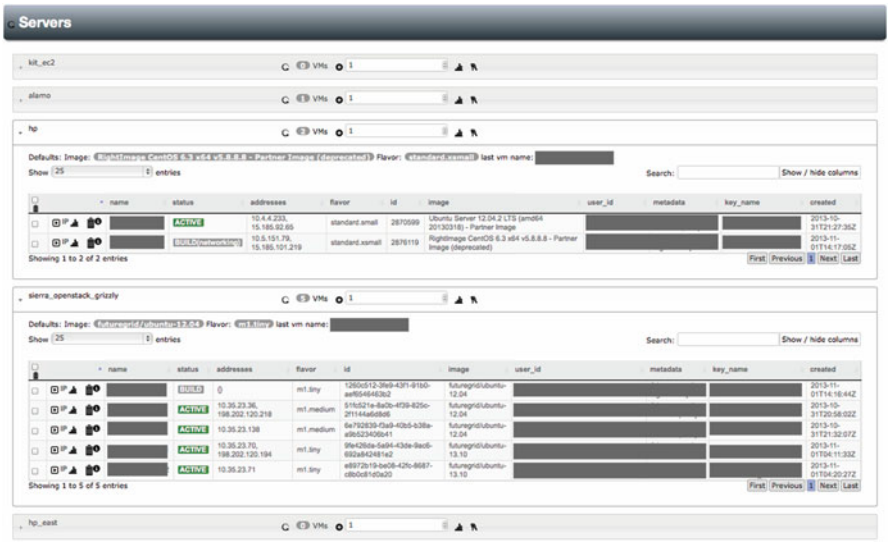


Fig. 21 Screenshot demonstrating how easy it is to manage multiple VMs across various clouds

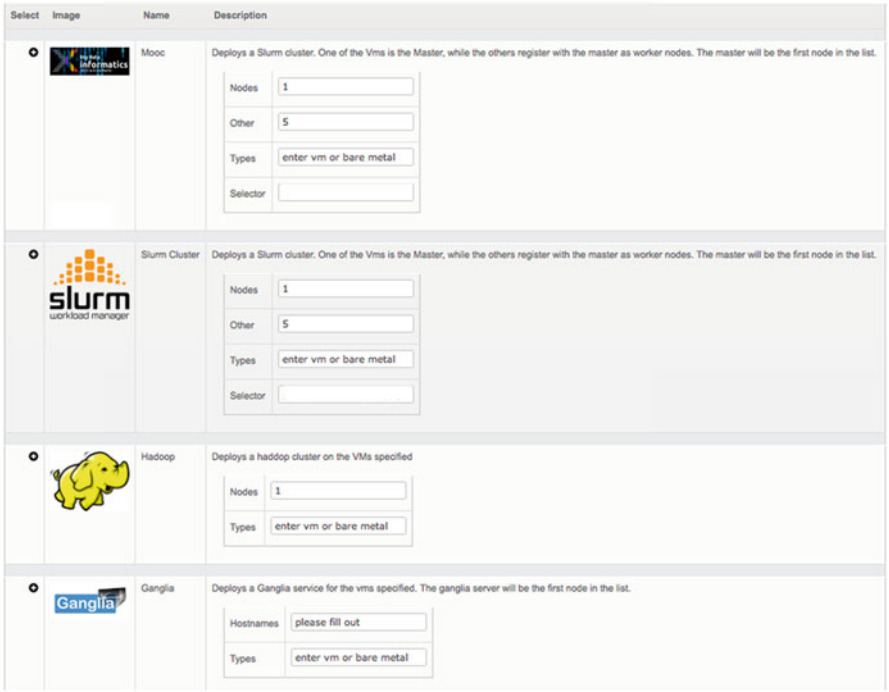


Fig. 22 One click deployment of platforms and sophisticated services that could even spawn multiple resources

register new clouds with the command shell, and have them show up also in the GUI that is managed through a hosted web server. Through this combination Cloudmesh can be seamlessly used as command shell or as GUI.

```
  ____ _                 _                                 _
 / ___| | ___  _   _  __| |_ __ ___   ___  ___| |__
| |   | |/ _ \| | | |/ _` | '_ ` _ \ / _ \/ __| '_ \
| |___| | (_) | |_| | (_| | | | | | |  __/\__ \ | | |
 \____|_|\___/ \__,_|\__,_|_| |_| |_|\___||___/_| |_|
=========================================================
                    Cloudmesh Shell

cm> help

Documented commands (type help <topic>):
========================================
EOF     defaults  graphviz   keys   metric   py        script  var
banner  dot2      help       label  open     q         storm   verbose
clear   edit      info       list   pause    quit      timer   version
cloud   exec      init       login  plugins  rain      use     vm
count   exp       inventory  man    project  register  user

Cloud Commands
==============
cloud     exp   inventory  list    rain      storm  vm    project
defaults  init  label      metric  register  user   keys
```

Fig. 23 The cloudmesh shell

Fig. 24 Listing the clouds

```
cm> cloud list
+--------------------------+
| Clouds                   |
+--------------------------+
| alamo                    |
| aws                      |
| azure                    |
| devstack_icehouse        |
| hp                       |
| hp_east                  |
| india_eucalyptus         |
| india_openstack_havana   |
| sierra_openstack_grizzly |
+--------------------------+
```

7 Summary

In this chapter, we have described FutureGrid and provided a general overview. In addition, we have analyzed the many projects executed on FutureGrid while paying attention on those requesting and using technologies relevant for big data analysis. Based on the discussion, it is clear that systems such as FutureGrid is extremely complex but provides to its users the benefit to offer multiple services within

the same infrastructure. This includes bare metal provisioning. Thus, performance experiments can not only be conducted on virtual machines, but on a variety of IaaS and PaaS environments. Moreover, these experiments can directly be compared to bare metal provisioned services. Hence, users can evaluate what impact such technologies have on their codes. Comparisons of different programming frameworks can be achieved and future activities in regards to efficiency and usability can be deducted. The lessons learned from FutureGrid are motivating a toolkit, Cloudmesh, that currently allows managing virtual machines on a variety of infrastructure as service frameworks. The easy deployment of sophisticated setups with one-click has been validated as part of an infrastructure designed for a MOOC. Furthermore the novel concept of shifting resources [29] between services to support services that need more resources is a significant contribution by Cloudmesh. Image management and creation under security restrictions [10] is furthermore an important aspect. We will continue to develop the Cloudmesh environment and make it available to our users.

Acknowledgements Some of the text published in this chapter is available form the FutureGrid portal. The FutureGrid project is funded by the National Science Foundation (NSF) and is led by Indiana University with University of Chicago, University of Florida, San Diego Supercomputing Center, Texas Advanced Computing Center, University of Virginia, University of Tennessee, University of Southern California, Dresden, Purdue University, and Grid 5000 as partner sites. This material is based upon work supported in part by the National Science Foundation under Grant No. 0910812 [11]. If you use FutureGrid and produce a paper or presentation, we ask you to include the references [11, 27] as well as this chapter. We like to thank Fugang Wang for the development of the framework that allowed us to produce the statistical data and Hyungro Lee for assisting in the creation of the data tables that lead to the creation of Figs. 11 and 12. Furthermore we like to thank Barbara O'Leary for proofreading this paper.

References

1. "Apache Hadoop Project." [Online]. Available: http://hadoop.apache.org
2. "Gartner's 2013 hype cycle for emerging technologies maps out evolving relationship between humans and machines," Press Release. [Online]. Available: http://www.gartner.com/newsroom/id/2575515
3. "Jira ticket system," Web Page. [Online]. Available: https://confluence.atlassian.com/display/JIRAKB/Using+JIRA+for+Helpdesk+or+Support
4. "Map reduce," Wikepedia. [Online]. Available: http://en.wikipedia.org/wiki/MapReduce
5. "Rt: Request tracker," Web Page. [Online]. Available: http://www.bestpractical.com/rt/
6. "Twister: Iterative mapreduce," Web Page. [Online]. Available: http://www.iterativemapreduce.org
7. W. Barth, *Nagios. System and Network Monitoring*, u.s. ed ed. No Starch Press, 2006. [Online]. Available: http://www.amazon.de/gp/redirect.html%3FASIN=1593270704%26tag=ws%26lcode=xm2%26cID=2025%26ccmID=165953%26location=/o/ASIN/1593270704%253FSubscriptionId=13CT5CVB80YFWJEPWS02
8. J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, Jan. 2008. [Online]. Available: http://doi.acm.org/10.1145/1327452.1327492

9. J. Diaz, G. von Laszewski, F. Wang, and G. C. Fox, "Abstract Image Management and Universal Image Registration for Cloud and HPC Infrastructures," in *IEEE Cloud 2012*, Honolulu, Jun. 2012. [Online]. Available: http://cyberaide.googlecode.com/svn/trunk/papers/12-cloud12-imagemanagement/vonLaszewski-12-IEEECloud2012.pdf
10. J. Diaz, G. von Laszewski, F. Wang, A. J. Younge, and G. C. Fox, "FutureGrid Image Repository: A Generic Catalog and Storage System for Heterogeneous Virtual Machine Images," in *Third IEEE International Conference on Coud Computing Technology and Science (CloudCom2011)*, Athens, Greece, 12/2011 2011, paper, pp. 560–564. [Online]. Available: http://cyberaide.googlecode.com/svn/trunk/papers/11-cloudcom11-imagerepo/vonLaszewski-draft-11-imagerepo.pdf
11. G. C. Fox, G. von Laszewski, J. Diaz, K. Keahey, J. Fortes, R. Figueiredo, S. Smallen, W. Smith, and A. Grimshaw, *Contemporary HPC Architectures*, draft ed., 2012, ch. FutureGrid - a reconfigurable testbed for Cloud, HPC and Grid Computing. [Online]. Available: http://cyberaide.googlecode.com/svn/trunk/papers/pdf/vonLaszewski-12-fg-bookchapter.pdf
12. Gregor, "Cloudmesh on Github," Web Page. [Online]. Available: http://cloudmesh.github.io/cloudmesh/
13. S. Krishnan and G. von Laszewski, "Using hadoop on futuregrid," Web Page, Manual, 2013. [Online]. Available: http://futuregrid.github.io/manual/hadoop.html
14. "The Network Impairments device is Spirent XGEM," 2012. [Online]. Available: http://www.spirent.com/Solutions-Directory/ImpairmentsGEM.aspx?oldtab=0&oldpg0=2
15. S. Krishnan, M. Tatineni, and C. Baru, "myHadoop - Hadoop-on-Demand on Traditional HPC Resources," Tech. Rep., 2011. [Online]. Available: http://www.sdsc.edu/~allans/MyHadoop.pdf
16. G. K. Lockwood, "myhadoop 2." [Online]. Available: https://github.com/glennklockwood/myhadoop
17. M. L. Massie, B. N. Chun, and D. E. Culler, "The Ganglia Distributed Monitoring System: Design, Implementation, and Experience," in *Journal of Parallel Computing*, April 2004.
18. J. Qiu, "Course: Fall 2013 P434 Distributed Systems Undergraduate Course." [Online]. Available: https://portal.futuregrid.org/projects/368
19. ——, "Spring 2014 CSCI-B649 Cloud Computing MOOC for residential and online students." [Online]. Available: https://portal.futuregrid.org/projects/405
20. L. Ramakrishnan, "FRIEDA: Flexible Robust Intelligent Elastic Data Management." [Online]. Available: https://portal.futuregrid.org/projects/298
21. "ScaleMP," 2012. [Online]. Available: http://www.scalemp.com/
22. S. Smallen, K. Ericson, J. Hayes, and C. Olschanowsky, "User-level grid monitoring with inca 2," in *Proceedings of the 2007 workshop on Grid monitoring*, ser. GMW '07. New York, NY, USA: ACM, 2007, pp. 29–38. [Online]. Available: http://doi.acm.org/10.1145/1272680.1272687
23. G. von Laszewski, "Cmd3," Github Documentation and Code. [Online]. Available: http://cloudmesh.futuregrid.org/cmd3/
24. ——, "Workflow Concepts of the Java CoG Kit," *Journal of Grid Computing*, vol. 3, pp. 239–258, Jan. 2005. [Online]. Available: http://cyberaide.googlecode.com/svn/trunk/papers/anl/vonLaszewski-workflow-taylor-anl.pdf
25. G. von Laszewski, J. Diaz, F. Wang, and G. C. Fox, "Comparison of Multiple Cloud Frameworks," in *IEEE Cloud 2012*, Honolulu, HI, Jun. 2012. [Online]. Available: http://cyberaide.googlecode.com/svn/trunk/papers/12-cloud12-cloudcompare/laszewski-IEEECloud2012_id-4803.pdf
26. G. von Laszewski, I. Foster, J. Gawor, and P. Lane, "A Java Commodity Grid Kit," *Concurrency and Computation: Practice and Experience*, vol. 13, no. 8–9, pp. 645–662, 2001. [Online]. Available: http://cyberaide.googlecode.com/svn/trunk/papers/anl/vonLaszewski-cog-cpe-final.pdf

27. G. von Laszewski, G. C. Fox, F. Wang, A. J. Younge, Kulshrestha, G. G. Pike, W. Smith, J. Voeckler, R. J. Figueiredo, J. Fortes, K. Keahey, and E. Deelman, "Design of the FutureGrid Experiment Management Framework," in *Proceedings of Gateway Computing Environments 2010 (GCE2010) at SC10*. New Orleans, LA: IEEE, Nov. 2010. [Online]. Available: http://cyberaide.googlecode.com/svn/trunk/papers/10-FG-exp-GCE10/vonLaszewski-10-FG-exp-GCE10.pdf
28. G. von Laszewski, M. Hategan, and D. Kodeboyina, *Workflows for E-science: Scientific Workflows for Grids*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2007, ch. Grid Workflow with the Java CoG Kit. [Online]. Available: http://cyberaide.googlecode.com/svn/trunk/papers/anl/vonLaszewski-workflow-book.pdf
29. G. von Laszewski, H. Lee, J. Diaz, F. Wang, K. Tanaka, S. Karavinkoppa, G. C. Fox, and T. Furlani, "Design of an Accounting and Metric-based Cloud-shifting and Cloud-seeding Framework for Federated Clouds and Bare-metal Environments," in *Proceedings of the 2012 Workshop on Cloud Services, Federation, and the 8th Open Cirrus Summit*, ser. FederatedClouds '12. New York, NY, USA: ACM, 2012, pp. 25–32.
30. G. von Laszewski and X. Yang, "Virtual cluster with slurm," Github repository. [Online]. Available: https://github.com/cloudmesh/cluster
31. A. J. Younge, R. Henschel, J. T. Brown, G. von Laszewski, J. Qiu, and G. C. Fox, "Analysis of Virtualization Technologies for High Performance Computing Environments," in *Proceedings of the 4th International Conference on Cloud Computing (CLOUD 2011)*. Washington, DC: IEEE, July 2011, pp. 9–16. [Online]. Available: http://cyberaide.googlecode.com/svn/trunk/papers/10-fg-hypervisor/10-fg-hypervisor.pdf

http://www.springer.com/978-1-4939-1904-8

Cloud Computing for Data-Intensive Applications
Li, X.; Qiu, J. (Eds.)
2014, VIII, 427 p. 180 illus., Hardcover
ISBN: 978-1-4939-1904-8