Chapter in 'Contemporary HPC Architectures'

Contents

1 FutureGrid - a reconfigurable testbed for Cloud, HPC and Grid Computing **1**

Geoffrey C. Fox, Gregor von Laszewski, Javier Diaz, Kate Keahey, Jose Fortes, Renato Figueiredo, Shava Smallen, Warren Smith, and Andrew Grimshaw

1.1 Overview . . . 2
1.1.1 Sponsor/Program background . . . 3
1.1.2 Historical Aspects of Clouds - Grids - HPC . . . 3
1.2 Hardware Resources . . . 4
1.3 Software Services . . . 5
1.3.1 Architecture Overview . . . 9
1.3.2 Access Services . . . 10
1.3.2.1 Access Service Demand . . . 11
1.3.2.2 Infrastructure as a Service . . . 11
1.3.2.3 Platform as a Service . . . 14
1.3.3 Management Services . . . 16
1.3.3.1 Dynamic Image Provisioning with RAIN . . 16
1.3.3.2 Monitoring, Information, and Performance Services . . . 23
1.3.4 Experiment and Project Management . . . 25
1.3.4.1 Interactive Experiment Management . . . 26
1.3.4.2 Workflow-Based Experiment Management . . 26
1.3.4.3 DevOps-Based Experiment Management . . 27
1.3.4.4 Rain-Based Experiment Management . . . 27
1.3.5 Operations Services . . . 27
1.3.6 Development Services . . . 28
1.3.7 Portal . . . 28
1.4 Service Deployment . . . 29
1.5 Applications using FutureGrid . . . 29
1.5.1 Privacy preserving gene read mapping using hybrid cloud . . . 30
1.5.2 SAGA on FutureGrid . . . 32
1.6 Optimizing MapReduce . . . 33
1.7 Sensor Cloud . . . 34
1.7.1 FutureGrid as a test bed for XSEDE (eXtreme Science and Engineering Discovery Environment) . . . 34

1.8 Educational Outreach 38
1.9 Operational Lessons. We have already learnt several unexpected lessons. 40

Bibliography **43**

Index **51**

Chapter 1

FutureGrid - a reconfigurable testbed for Cloud, HPC and Grid Computing

Geoffrey C. Fox

Pervasive Technology Institute, Indiana University
2729 E 10th St., Bloomington, IN 47408, U.S.A.

Gregor von Laszewski

Pervasive Technology Institute, Indiana University
2729 E 10th St., Bloomington, IN 47408, U.S.A.
E-mail: laszewski@gmail.com

Javier Diaz

Pervasive Technology Institute, Indiana University
2729 E 10th St., Bloomington, IN 47408, U.S.A.

Kate Keahey

Argonne National Laboratory, Argonne, IL

Jose Fortes

University of Florida, Gainesville, FL

Renato Figueiredo

University of Florida, Gainesville, FL

Shava Smallen

San Diego Super Computing Center, San Diego, CA

Warren Smith

Texas Advanced Computing Center, Austin, TX

Andrew Grimshaw

University of Virginia, Charlottesville, VA

1.1 Overview 2
1.1.1 Sponsor/Program background 3
1.1.2 Historical Aspects of Clouds - Grids - HPC 3
1.2 Hardware Resources 3
1.3 Software Services 4

1.3.1 Architecture Overview 8
1.3.2 Access Services 10
1.3.2.1 Access Service Demand 11
1.3.2.2 Infrastructure as a Service 11
1.3.2.3 Platform as a Service 14
1.3.3 Management Services 16
1.3.3.1 Dynamic Image Provisioning with RAIN 16
1.3.3.2 Monitoring, Information, and Performance Services 23
1.3.4 Experiment and Project Management 25
1.3.4.1 Interactive Experiment Management 26
1.3.4.2 Workflow-Based Experiment Management 26
1.3.4.3 DevOps-Based Experiment Management 26
1.3.4.4 Rain-Based Experiment Management 27
1.3.5 Operations Services 27
1.3.6 Development Services 28
1.3.7 Portal 28
1.4 Service Deployment 28
1.5 Applications using FutureGrid 29
1.5.1 Privacy preserving gene read mapping using hybrid cloud 30
1.5.2 SAGA on FutureGrid 32
1.6 Optimizing MapReduce 33
1.7 Sensor Cloud 33
1.7.1 FutureGrid as a test bed for XSEDE (eXtreme Science and Engineering Discovery Environment) 34
1.8 Educational Outreach 38
1.9 Operational Lessons. We have already learnt several unexpected lessons. 40
Acknowledgement 41

xf

1.1 Overview

The FutureGrid project [vLFW+10] mission is to enable experimental work that advances:

- innovation and scientific understanding of distributed computing and parallel computing paradigms,
- the engineering science of middleware that enables these paradigms,
- the use and drivers of these paradigms by important applications, and,
- the education of a new generation of students and workforce on the use of these paradigms and their applications.

The implementation of the mission includes

- distributed flexible hardware with supported use,

- identified Infrastructure as a Service (IaaS) and Platform as a Service (PaaS) core software with supported use,
- a growing list of software from FutureGrid partners and users, and
- educational outreach. ¡

Thereby the FutureGrid project provides a capability that makes it possible for researchers to tackle complex research challenges in computational and computer and computational science related to the use and technology of High Performance Computing (HPC) systems, Grids [FK99], and clouds [AFG+09, KH11]. Topics range from programming models, scheduling, virtualization, middleware, storage systems, interface design and cybersecurity, to the optimization of Grid-enabled andxfgg cloud-enabled computational schemes for researchers in astronomy, chemistry, biology, engineering, atmospheric science and epidemiology.

1.1.1 Sponsor/Program background

FutureGrid [wwwb] is sponsored by the National Science Foundation under Grant No. 0910812 [nsf09] to Indiana University for "FutureGrid: An Experimental, High-Performance Grid Test-bed." FutureGrid forms part of NSF's national high-performance cyberinfrastructure XSEDE [xse12]. It increases the capability of the XSEDE to support innovative computer science research requiring access to lower levels of the grid software stack, the networking software stack, and to virtualization and workflow orchestration tools as well as new programming models like MapReduce [DG08]. As it supports interactive use, it is well suited for testing and supporting distributed system and scientific computing classes. Education and broader outreach activities include the dissemination of curricular materials on the use of FutureGrid, pre-packaged FutureGrid virtual machines (appliances) configured for particular course modules, and educational modules based on virtual appliance networks and social networking technologies [SBC+03, BFLK11].

Partners in the FutureGrid project include U. Chicago, U. Florida, San Diego Supercomputer Center - UC San Diego, U. Southern California, U. Texas at Austin, U. Tennessee at Knoxville, U. of Virginia, Purdue I., T-U. Dresden, and Grid5000 [imp12b]. These cover hardware, software and benchmarking.

1.1.2 Historical Aspects of Clouds - Grids - HPC

One of the unique aspects of FutureGrid is that it is not nly targeted towards the use of High Performnce computing but also towards the integration of Clouds and Grids. The concepts from these areas are strongly interconnected and can be put in a historical context as shown in Figure 1.1.

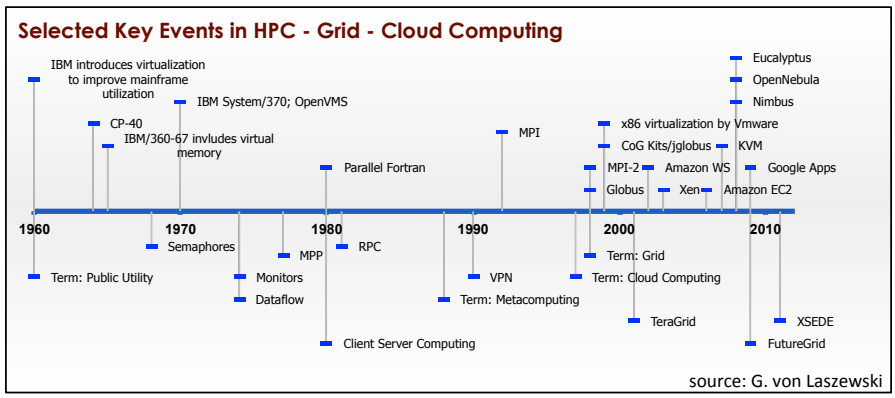

FIGURE 1.1: Selected key events in HPC, Grid, and Cloud computing

1.2 Hardware Resources

FutureGrid (FG) is a national-scale Grid and cloud test-bed facility that includes a number of computational resources at distributed locations (see Table 1.1). The FutureGrid network is unique and can lend itself to a multitude of experiments specifically for evaluating middleware technologies and experiment management services [vLFW+10]. This network can be dedicated to conduct experiments in isolation, using a network impairment device for introducing a variety of predetermined network conditions. Figure 1.2 depicts the network infrastructure, Table 1.1 lists computational resources and Table 1.2 the storage resources. All network links within FutureGrid are dedicated (10GbE lines for all but to Florida, which is 1GbE), except the link to TACC. The significant number of distinct systems within FutureGrid provide a heterogeneous distributed architecture and are connected by high-bandwidth network links supporting distributed system research. One important feature to note is that some systems can be dynamically provisioned, e.g. these systems can be reconfigured when needed by special software that is part of FutureGrid with proper access control by users and administrators. A Spirent H10 XGEM Network Impairment emulator [imp12c] co-located with the core router [jun12], a central resource to introduce network latency, jitter, loss, and errors to network traffic within FutureGrid.

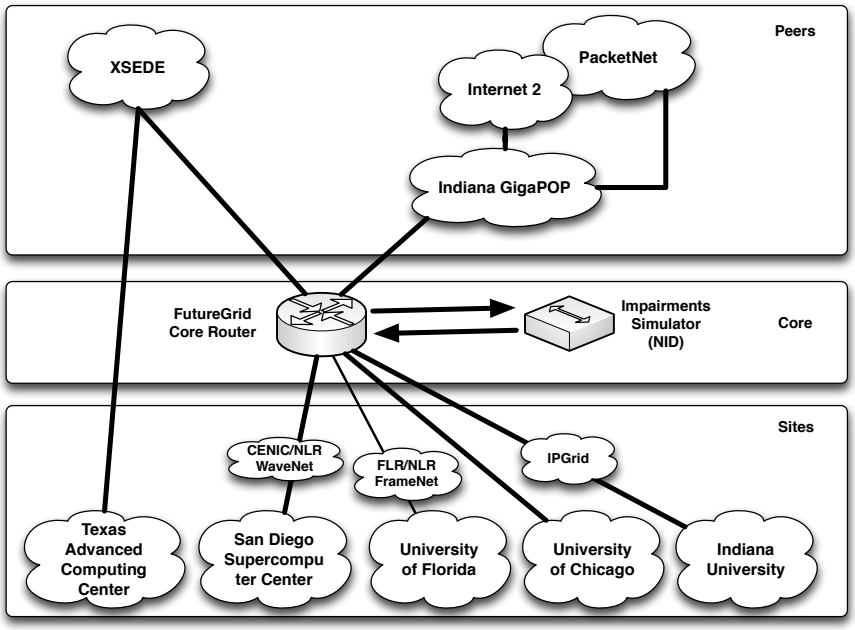

FIGURE 1.2: Network Infrastructure of FutureGrid

1.3 Software Services

In order for FG to operate we have to conduct a number of activities related to developing, deploying and supporting the software for FG.

The goal set by FG is to provide a "an experimental Grid, Cloud, and HPC testbed". This goal has naturally a direct impact on our software design, architecture, and deployment. Hence, we revisit some of the elementary requirements influenced by the user community, various access models and services, and the desire to be able to conduct a variety of reproducible experiments. These requirements include:

Support a Diverse User Community. As part of our initial investigations, we have identified a number of different user communities that will benefit from a testbed such as FG. Naturally, the desire to support these communities governs our software design and the access to FG in general. We support the following communities:

- *Application developers* that investigate the use of software and services provided by FG;

TABLE 1.1: Current Compute Resources of FutureGrid as of April 2012

Name	System Type	Nodes	CPU's	Cores	TFLOPS	RAM GB	Site
india	IBM iDataplex	128	256	1024	11	3072	IU
hotel	IBM iDataplex	84	168	672	7	2016	UC
sierra	IBM iDataplex	84	168	672	7	2688	SDSC
foxtrot	IBM iDataplex	32	64	256	3	768	UF
alamo	Dell PowerEdge	96	192	768	8	1152	TACC
xray	Cray XT5m	1	168	672	6	1344	IU
bravo	HP Proliant	16	32	128	1.7	3072	IU
delta	GPU Cluster	16	32	192	TBD	3072	IU
				14336*			
Total		**457**	**1080**	**4384**	**>43.7**	**17184**	
				14336*			

*GPUS

TABLE 1.2: Storage Resources of FutureGrid as of April. 2012

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

- *Middleware developers* that investigate the development of middleware and services for Cloud and Grid computing;
- *System administrators* that investigate technologies that they wish to deploy into their own infrastructure;
- *Educators* that like to expose their students to software and services to Cloud and Grid computing technologies as offered on the FG; and
- *Application users* that like to test out services developed by application and middleware developers.

To support this diverse community we have identified a number of key requirements we will be focusing on as part of FutureGrid.

Support for Shifting Technology Base. One of the observations that motivated FG is the rapidly developing technologies in the cloud and Grid that may have profound impact on how we develop the next generation scientific applications keeping these new developments in mind. The introduction of virtualization [AA06, CB09, cC05], Infrastructure as a Service (IaaS), and Platform as a Service (PaaS) paradigms [LKN+09, KH11] calls for the ability to have access to software tools and services that allow a comparison of these paradigms with traditional HPC methodologies.

Support a Diverse Set of Interface Methods. Based on this technology shift and the interest posed by the various user communities to have easy interfaces to a testbed, we need to develop, as part of our software activities, appropriate interface tools and services. The interfaces include command line tools, APIs, libraries and services. In addition, many users would like access to these new technologies through portals or GUIs.

Support a Diverse Set of Access Methods. While in previous decades the focus has been to provide convenient libraries, tools, and Web services we currently see an expansion into infrastructure and platform as services. Thus, a new generation of tools and services are provided as abstractions to a higher level of services, potentially replacing the traditional OS. Thus, we will not only offer access to Infrastructure as a Service (IaaS) framework, but we will also invest in providing PaaS endpoints, thereby allowing access to a new kind of abstraction.

Support a Diverse Set of Access Services. ue to the rapid development of new tools, services, and frameworks within the Grid and cloud communities, it is important to facilitate a multitude of such environments. This includes access to IaaS frameworks such as Nimbus [wwwc], Eucalyptus [wwwf], OpenNebula [wwwg], OpenStack [wwwh]; PaaS frameworks such as Hadoop [wwwa]; and additional services and tools like Unicore [OR02] and Genesis II [imp12a], that are provided and supported by the FG team members. Hence, users will have the ability to investigate a number of different frameworks as part of their activities on FG.

Support of Traditional Services. We provide a number of additional services that users are accustomed to. This includes High Performance Computing (HPC) services [Com12], but also access to backup and storage. Naturally, we provide services for user support as part of a portal with access to information including a ticket system.

Support for Persistent Services and Endpoints. One of the potential assets of FG is the ability to expose a number of students and practitioners to the new frameworks offered. However, the entry to set up such systems may have to be low in order to interest others in using such technologies. Hence, it is important to offer a number of persistent services and endpoints of such frameworks. Furthermore, we must make it easy for the teachers and administrators of FG to manage membership and access rights to such endpoints. In addition, we are interested in providing a number of "standard" images for educational purposes that can be used for teaching about particular aspects.

Support for Raining/Dynamic Provisioning. As we are not only interested in offering a single pre-installed OS or IaaS framework, we must provide additional functionality to reassign service nodes to a particular framework. To support this implicit move of resources to different

services, we we need to offer dynamic provisioning within FG not only within an IaaS framework, such as Nimbus, Eucalyptus or OpenStack, but also any OS we choose. Furthermore this concept can be expanded to provision additional platforms and services. Hence, we use the term "raining" instead of just dynamic provisioning to indicate that we strive to dynamically provision not only on the OS level, but also on the service and platform level [vLFW+10]. This combination will will allow us to provide efficient assignment of resources to services governed by user needs. Thus, if there is no demand for running Eucalyptus staged images, the resources devoted to the Eucalyptus cloud can be de-registered and assigned to a different service. An additional aspect of our "rain" tool that we are developing is to specify the mapping onto specific resources, allowing us to compare services on the same hardware.

Support for a Viral User Contribution Model. User contributions are possible at several levels. First, the development of shared images that are instantiated as part of an IaaS framework. Second, the development of middleware, either in the IaaS or PaaS models, that can be rained onto FG. Third, the creation of workflows that utilize a combination of the services offered as part of sophisticated experiment workflows, which we will explain in more detail later. Fourth, through the contribution of educational material.

One of the important features to recognize is that FG distinguishes itself from current available systems. This includes traditional compute centers such as XSEDE [xse12], but also well known IaaS offerings, such as Amazon [Amaa].

In contrast to Amazon, we provide alternatives to the IaaS framework, but the biggest benefit stems from two unique features of FG. In FG we intend to allow the mapping of specific resources as part of the service instantiation. Thus, we can measure more realistically performance impacts of the middleware and the services developed by the FG testbed user. Furthermore, we allow authorized users a much greater level of access to resources by allowing the creation of images that can not only be placed in a virtual machine (VM) but also be run on the "bare" hardware.

A big distinction between XSEDE and FG is that FG provides a more variable software stack and services. Traditionally, supercomputing centers that are part of XSEDE focus on large scale high performance computing applications while providing a well defined software stack. Access is based on job management and traditional parallel and distributed computing concepts. Virtual machine staging on XSEDE has not yet deemed to be a major part of its mission. FG is more flexible in providing software stacks to be dynamically adapt by the users.

1.3.1 Architecture Overview

To support our requirements we have devised a flexible software architecture enabling us to gradually introduce and expand components in support of our mission. We distinguish the following components:

Fabric: The Fabric layer contains the hardware resources, including the FG computational resources, storage servers, and network infrastructure including the network impairment device.

Development and Support Fabric/Resources: Additional resources are set aside that are helping us with the development and support of operational services. This includes servers for portals, ticket systems, task management systems, code repositories, a machine to host an LDAP [KV04] server and other services. It is important to recognize that such services should not be hosted on the "cluster" resources that constitute the main FG Fabric as an outage of the cluster would affect important operational services.

Operations Services: In order to effectively communicate and conduct development effort, the following elementary services have been provided: a website, a development wiki, a task management system to coordinate the software development tasks and a ticket system. In addition, we need to provide security and accounting services to deal with authentication, authorization and auditing.

Base Software and Services: The FG Base services contain a number of services we rely on when developing software in support of the FG mission. This includes Software that is very close to the FG Fabric and includes tools like Moab [Com12], xCAT [wwwj], and also the base OS. This category of services will enable us to build experiment management systems utilizing dynamic provisioning.

Management Services: The management services are centered around FG experiments and the overall system integration, including information services and raining/dynamic provisioning (see Section 1.3.3) software stacks and environments on the Fabric.

Access Services: FG user services contain variety of services. They include IaaS, PaaS, SaaS, and classical Libraries that provide a service as an infrastructure to the users such as accessing MPI [SOHL+98] and others (see Section 1.3.2).

User Contributed Services (as part of additional Services): The architecture image does not explicitly distinguish user contributed services. It is important to note that user contributions take place on many different access levels. This is supported by our architecture by allowing the creation, distribution, reuse and instantiation of user contributed

software as part of services or experiments within FG. Thus, we expect that the FG User Contributed Services will grow over time while enhancing areas that we have not explicitly targeted ourselves. Instead, we provide mechanisms for community users to integrate their contributions into FG offered services. The only difference to these services may be the level of support offered in contrast to other FG services.

Together these components build our layers architecture view as depicted in Figure 1.3.

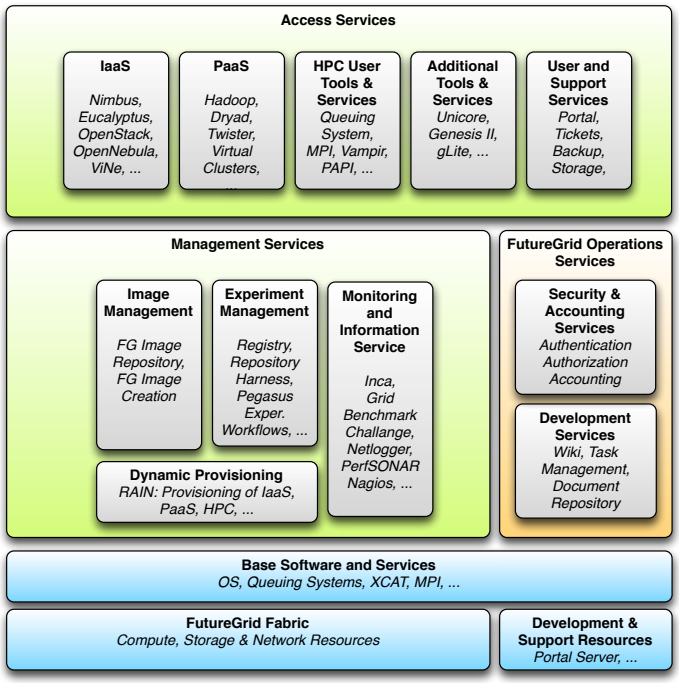

FIGURE 1.3: FutureGrid Software Architecture

1.3.2 Access Services

Next we will be focusing our attention towards the access services. As part of the access services, we distinguish the following areas:

- PaaS (Platform as a Service): Delivery of a computing platform and solution stack;
- IaaS (Infrastructure as a Service): Deliver a compute infrastructure as a service;

- Grid: Deliver services to support the creation of virtual organizations contributing resources;
- HPCC (High Performance Computing Cluster): Traditional high performance computing cluster environment; and
- User and Support Services: Delivery of services to enable user support.

1.3.2.1 Access Service Demand

The question arises which services should we offer in FutureGrid? To answer this question we have we identified user demand within the community as part of our project application process within FutureGrid. At registration time of a project, the project owner is presented with a number of choices of technologies that are most useful for their project or they desire to have access to. Project owners were able to choose multiple items. We obtained the following result while analyzing the requests from project owners: (a) Nimbus: (53.2%), (b) Eucalyptus: (50.8%), (c) Hadoop: (37.3%), (d) High Performance Computing Environment: (35.7%), (e) MapReduce: (33.3%), (f) Common TeraGrid Software Stack: (27%), (g) Genesis II: (16.7%), (h) Twister: (15.9%), (i) OpenStack: (12.7%), (j) OpenNebula: (11.1%), (k) Unicore 6: (10.3%), (l) gLite: (9.5%)

Please note that the the data is only collected for each project owner and does not contain information gathered by members of projects or other FutureGrid users. We observed over time only slight changes in the request to these services. However, most recently we have seen significant increased demand for OpenStack. As part of this monitoring activity, we intend to be flexible in what we offer on FutureGrid and work with the community to strive towards fulfilling services needs that fall within our project goals. We realize that gathering this information may be biased as we gather the information from our current set of users and therefore we are also monitoring the community actively (see Figure 1.4). This helps us attracting new users and adopt our strategies towards community needs [vLDWF12].

1.3.2.2 Infrastructure as a Service

In contrast to the traditional offering of supercomputer centers, FutureGrid provides a variety of Infrastructure as a Service (IaaS) frameworks on its resources. IaaS allows to abstract the physical hardware and offer users instead access to virtual machines that are than mapped onto the hardware. Currently, one of the special features of FutureGrid is to provide not only one IaaS framework, but several of them.

The reason why FG provides multiple IaaS frameworks is based on the fact that any of the IaaS frameworks are under heavy development. Features are added, and performance changes based on versions and deployment strategies. Thus it is important to offer a variety of IaaS frameworks to a assist the users

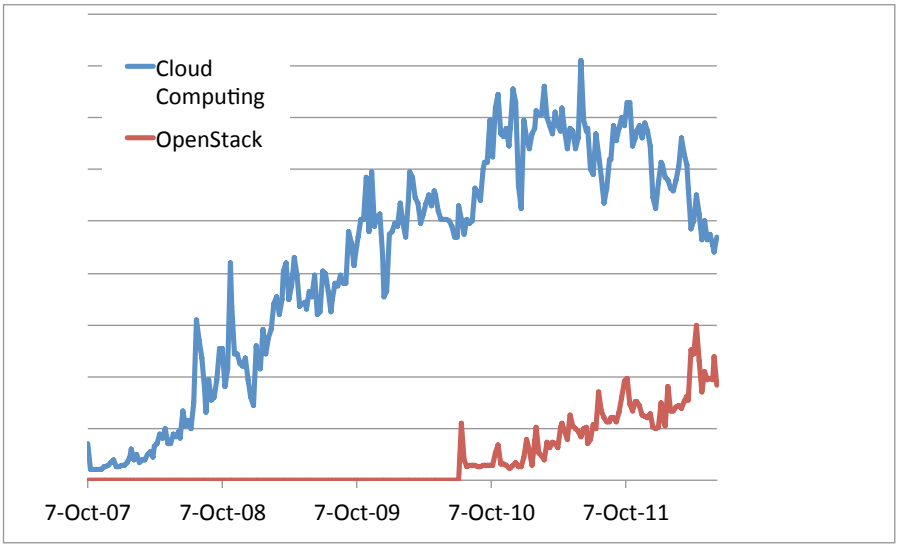

FIGURE 1.4: Google Trends shows popularity of OpenStack to be significantly rising

evaluating and identifying the IaaS and deployment strategies best suited for their applications. While on the surface client tools seem to offer similar capabilities, the difference may be in how such frameworks are implemented and how they perform and scale as part of an application integration. The current IaaS frameworks that are offered on FutureGrid include Nimbus [wwwc], OpenStack [wwwh] and Eucalyptus [wwwf]. Some of our own project also use OpenNebula [wwwg] but we have not yet made it publicly available.

Nimbus. The Nimbus project [wwwc] consists of an IaaS (Nimbus Infrasturcuture) and a PaaS component (Nimbus Platform).

Nimbus Infrastructure provides both compute and storage cloud compatible with Amazon Web Service's Elastic Compute Cloud (EC2) [Amaa] and Simple Storage Service (S3) [Amab], respectively. Nimbus targets features of interest to the scientific community such as support for proxy credentials, enhanced scheduling options, and fast deployment of large virtual clusters. These features make it easy for scientific projects to experiment with FutureGrid Nimbus installations. Further, both Nimbus compute (Workspace Service [KFF+05]) and storage (Cumulus [BFLK10]) cloud components are provided as a high-quality, highly configurable and extensible open source implementation which allows students and scientists to modify them in order to experiment with new capabilities as has been done in [MKF11, RTM+10]. The FutureGrid Nimbus deployment contains of four independent clouds on Hotel, Sierra, Foxtrot and Alamo, allowing research in the development of heterogeneous cloud environments.

Nimbus Patform [KF08, BFLK11] is an integrated set of tools that allow users to provision resources across multiple infrastructure clouds, create virtual clusters across those clouds, as well as scale to demand. Coordinated deployment of complex resource configuration across multiple clouds is accomplished with Nimbus cloudinit.d [BFLK11] which allows users to develop a launchplan for such complex deployment once and then execute it many times. The Nimbus context broker [BFLK11] coordinates secure creation of "one click" virtual clusters, potentially distributed over multiple clouds. These services allow users to combine resources provisioned from Nimbus, OpenStack, and Amazon Web Services clouds and are designed to be extended to support other IaaS providers. We are in the process of developing multi-cloud autoscaling services which will further facilitate access to FutureGrid cloud resources.

Eucalyptus. Eucalyptus [wwwf] is a popular IaaS framework that has been commercialized since this project started. Our current deployment contains two independent services on India and Sierra allowing research in the development of heterogeneous cloud environments. Performance comparisons of Eucalyptus 2 to Openstack and Eucalyptus 3 have shon that it is no longer competitive. Hence we have recently updated from Eucalyptus 2 to Eucalyptus 3. The ability to conduct such performance experiments on the same resources to gather this information is a significant strength of FutureGrid. Our results were presented at the Eucalyptus users group meeting in 2012 and positively received not only by the community but also by the Eucalyptus team as it gave insight in how our user community uses the various IaaS frameworks and can compare them with each other.

OpenStack. Most recently OpenStack [wwwh] has been introduced to the community. It is driven by the community and has received a significant community backing not only by the academic, but also by the industrial community. The quality of the code base is rapidly improving. By now OpenStack has released a quite stable set of components that can be used to build significantly large clouds. The recent OpenStack user's community meeting was attended by 1000 users. This in addition to the rising trend (see Figure 1.4) proves that FutureGrid must support OpenStack on its resources.

ViNe: User Level Virtual Networks. ViNe [TF06] is a project developed at University of Florida that implements routing and other communication mechanisms needed to deploy a user-level virtual network. ViNe is particularly appealing for cloud computing because it allows the establishment of wide-area virtual networks supporting symmetric communication among public and private network resources (even when they are behind firewalls), does not require changes to either the physical network or the OS of machines, and has low virtualization overheads. ViNe can provide communication among FutureGrid and external resources (including those with private IP addresses) without the need to reconfigure the (FutureGrid) physical network infrastructure.

In the first phase of FG, ViNe efforts focused on deploying and demonstrating overlay network capabilities. In the largest experiment, ViNe connected a

virtual cluster (launched through Nimbus) across 3 FG (sierra, foxtrot, and hotel) and 3 Grid'5000 (Rennes, Lille, and Sophia) sites [imp12b]. The virtual cluster consisted of 750 VMs (1500 cores), and executed BLAST on Hadoop (CloudBLAST) with speedup of up to 870X [MTF09].

In the second phase, ViNe efforts focused on adding management capabilities to the existing code. The goal is to make it easy for FG users to configure and operate ViNe software, without the needed overlay networks expertise.

In the third phase, building on the management capabilities implemented previously, high level management services (e.g., end-to-end QoS, overlay network performance self-optimization, recovery in the presence of faults, etc) will be developed.

1.3.2.3 Platform as a Service

In addition to the IaaS, FutureGrid offers also the ability to provide Platforms as a Service to the users. Platforms typically include a well defined solution stack such as a customizes operating system, a programming language execution environment, databases, as well as platforms related to data analysis, such as map/reduce, Grid Computing, and even High Performance Computing (HPC). Once committed to a platform, users rely on the platform to be deployed and develop their applications against such platforms. As each of these platforms could be differently installed and managed it is sometimes beneficial to be able to install them in a different fashion. Thus although we have provided a number of default implementations, users are typically able to assemble their own platforms to increase performance while targeting available resources for specific application performance characteristics.

MapReduce. Within FutureGrid we provide a number of ways on how users can access MapReduce [DG08]. First we allow users to use the Hadoop-based Map/Reduce platform [wwwa] hosted on bare metal. However we also allow users to stage their own personalized versions of Hadoop as to allow modifications and research to take place as part of improvements to the Hadoop code base. In addition we also provide Hadoop as part of our dynamic provisioning service in order to simplify performance experimentations.

Hadoop is a very popular PaaS that provides users with the map/reduce framework. In addition to installing Hadoop on systems, FutureGrid has also deployed myHadoop [KTB11], a set of scripts developed by SDSC that makes it easy to submit Hadoop jobs through the FutureGrid batch queue systems. It is also easy to customize and allows users to make their own copy and adjust default Hadoop settings or to specify an alternate Hadoop implementation. This is important, as some users may want to experiment with modified versions of Hadoop.

High Performance Computing Services. To emphasize high-level platforms for High Performance computing and their special role, we have added a separate category for them in our architecture as depicted in Figure 1.3. As we are part of XSEDE some of our users have the need to test out software

that will later on be deployed in XSEDE. Hence providing regular HPC services such as access to a queuing system and being able to run MPI programs is possible. In addition to such bare metal services, we have also developed examples on how such queuing systems can be set up via a single command-line tool in a cloud environment on for example OpenStack. This is important for future activities where users may experiment with queueing strategies and federated data centers while simulating an XSEDE like infrastructure in the Cloud.

Grid Services. FutureGrid also offers a number of Grid services including access to Unicore, and Genesis II. Not surprisingly the demand for Globus was relatively low, as they have been available for some time on XSEDE and the users of such services seem to utilize the far larger XSEDE resources. Most of the demand for Globus stems form the workflow community that tries to also integrate FutureGrid into their workflows while relying on Grid services, as well as the interoperability research community that investigates tools to increase interoperability within Grid environments.

More interestingly for our community is the Globus Provision tool [glo12] for deploying fully configured Globus environments within Cloud environments. It is designed to be simple to use, and will allow you to deploy common Globus services, such as GridFTP and GRAM, in just minutes. Globus Provision will also take care of generating user accounts, certificates, and setting up auxiliary services. Globus Provision can deploy these services in any combination you need. For example, you could deploy a single GridFTP server, a Condor pool [con] with ten worker nodes and a GRAM server, or 30 GridFTP servers to teach a tutorial where each student needs their own GridFTP server to play around with. Once these services are deployed, you can dynamically add and remove software, hosts, and user accounts. We have made the current version of Globus provisioning available on FutureGrid, but have not yet received much feedback about its functionality and usage.

Symmetric Multiprocessing. Symmetric Multiprocessing (SMP) has been a fundamental driver in HPC, furthering the availability of parallel processing architectures to commodity multi-core technology. While this multi-core age has continued Moore's Law, the growth rate of cores has been lackluster at best. Cache coherent Non-Uniform Memory Access (ccNUMA) architectures, specifically based on the newest x86 processors, have the ability to turn a commodity cluster into a single, large-scale supercomputer. These ccNUMA machines provide large-scale multiprocessing and relative ease of use for data and compute intensive scientific applications, but the costs associated with such supercomputers make them prohibitively expensive for the vast majority of potential users.

Recently, virtualization has allowed for the ability to abstract hardware in order to create a virtualized SMP machine using commodity hardware. One such implementation, vSMP by ScaleMP Inc, provides such an experience [sca12]. However, as it is a virtualized access to SMP it is important to measure performance impact on applications. A deployment within FutureGrid allows such

analysis. Experiments conducted on futureGrid using HPCC benchmarks show only a 4-6% drop in efficiency when compared to native cluster performance.
PAPI and Vampir To support the development of performance based experimentation on FutureGrid we have deployed a number of tools for the user. This includes Vampir [vam12] and PAPI.
PAPI is an acronym for Performance Application Programming Interface [pap12]. The PAPI Project is being developed at the University of Tennessee's Innovative Computing Laboratory in their Computer Science Department. This project was created to design, standardize, and implement a portable and efficient API (Application Programming Interface) to access the hardware performance counters found on most modern microprocessors. PAPI is at this time enhanced to be able to work also in virtual machines.
Vampir provides a manageable framework for analysis, which enables developers to quickly display program behavior at any level of detail. Detailed performance data obtained from a parallel program execution can be analyzed with a collection of different performance views. Intuitive navigation and zooming are the key features of the tool, which help to quickly identify inefficient or faulty parts of a program code. Vampir implements optimized event analysis algorithms and customizable displays which enable a fast and interactive rendering of very complex performance monitoring data.

1.3.3 Management Services

1.3.3.1 Dynamic Image Provisioning with RAIN

Cloud computing has become an important driver for delivering infrastructure as a service (IaaS) to users with on-demand requests for customized environments and sophisticated software stacks. As we support a number of different IaaS frameworks we have to consider the following issues:

1. Resources need to be managed within each IaaS
2. Resources have to be assigned to each IaaS
3. Sharing of resources and reallocating them is an integral part of FutureGrids services.

Within the FutureGrid project, we are devising a framework that allows this level of customization for administrators and users. The FutureGrid architecture, depicted in Figure 1.3, shows the two components, namely *Image Management* and *Dynamic Provisioning*, that are tightly interwoven to allow users to dynamically provision images on bare-metal and virtualized infrastructures.
Image management is a key component in any modern compute infrastructure, regardless if used for virtualized or non-virtualized resources. We distinguish a number of important processes that are integral part of the life-cycle management of images. They include (a) image creation and customization,

(b) sharing the images via a repository, (c) registering the image into the infrastructure, and (d) image instantiation (see Figure 1.5). The problem of targeting not one, but multiple infrastructures amplifies the need for tools supporting these processes. Without them, only the most experienced users will be able to manage them under great investment of time.

Our design targets an end-to-end workflow to support users in creating abstract image management across different infrastructures easily [DvLWF12, DYvL+11]. This includes images for Eucalyptus, Nimbus, OpenStack, OpenNebula, Amazon, and bare-metal. To summarize the idea behind our design, we prefer users to be able to specify a list of requirements such as an OS, an architecture, software, and libraries in order to generate a personalized abstract image. This image is generic enough that through manipulations it can be adapted for several IaaS or HPC infrastructures with little effort by the users. It will support the management of images for Nimbus [wwwc], Eucalyptus [wwwf], OpenStack [wwwh], and bare-metal HPC infrastructures as they are either already deployed in FG or going to be deployed as in the case of OpenNebula [wwwg].

By supporting this image management workflow, our framework eases the use of IaaS and HPC frameworks and infrastructures for performance experiments based on abstract image management and uniform image registration. Consequently, users can build their own customized environments very easily. The complex processes of the underlying infrastructures are managed by our sophisticated software tools and services. Besides being able to manage images for IaaS frameworks, we also allow the registration and deployment of images onto bare-metal by the user. This level of functionality is typically not offered in a HPC infrastructure. However, our approach provides users with the ability to create their own environments changing the paradigm of administrator-controlled dynamic provisioning to user-controlled dynamic provisioning. Thus, users obtain access to a testbed with the ability to manage state-of-the-art software stacks that would otherwise not be supported in typical compute centers. Security is also considered by vetting images before they are registered in an infrastructure.

The capabilities provided by our image management framework are advantageous to support repeatable performance experiments across a variety of infrastructures. To support a modular design we have devised a component for each process. This includes an image generation component to create images following user requirements, an image repository component to store, catalog and share images, and an image registration component for preparing, uploading and registering images into specific infrastructures such as HPC or different clouds.

As we can see in Figure 1.5, the architecture includes a convenient separation between client and server components for allowing users to easily interact with the hosted services that manage our processes. Our design allows users to access to the various processes via a python API, a REST service [Fie00], a convenient command line shell, as well as a portal interface. The image

management server has the task to generate, store, and register the images with the infrastructure. The image management server also interfaces with external services, such as configuration management services to simplify the configuration steps, authentication and authorization, and a service to verify the validity of an image including security checks.

One important feature in our design is how we are not simply storing an image but rather focusing on the way an image is created through abstract templating. Thus, it is possible at any time to regenerate an image based on the template describing the software stack and services for a given image. This enables us also to optimize the storage needs for users to manage many images. Instead of storing each image individually, we could just store the template or a pedigree of templates used to generate the images.

To aid storage reduction, our design includes data to assist in measuring usage and performance. This data can be used to purge rarely used images, while they can be recreated on-demand by leveraging the use of templating. Moreover, the use of abstract image templating will allow us to automatically generate images for a variety of hypervisors and hardware platforms on-demand. Autonomous services could be added to reduce the time needed to create images or deploy them in advance. Reusing images among groups of users and the introduction of a cache as part of the image generation will reduce the memory footprint or avoid the generation all together if an image with the same properties is already available.

Image Generation. The image generation provides the first step in our image management process allowing users to create images according to their specifications. As already mentioned, the benefit of our image generation tools and services is that we are not just targeting a single infrastructure type but a range of them.

The process is depicted in Figure 1.6. Users initiate the process by specifying their requirements. These requirements can include the selection of the OS type, version, architecture, software, services, and more. First, the image generation tool searches into the image repository to identify a base image to be cloned, and if there is no good candidate, the base image is created from scratch. Once we have a base image, the image generation tool installs the software required by the user. This software must be in the official OS repositories or in the FG software repository. The later contains software developed by the FG team or other approved software. The installation procedure can be aided by Chef [CHEa], a configuration management tool to ensure the software is installed and configured properly. After updating the image, it is stored in the image repository and becomes available for registration into one of the supported infrastructures. Our tool is general to deal with installation particularities of different operating systems and architectures.

One feature of our design is to either create images from scratch or by cloning already created base images we locate in our repository.

In case we create an image from scratch, a single user identifies all specifications and requirements. This image is created using the tools to bootstrap

images provided by the different OSes, such as yum for CentOS and deboostrap for Ubuntu. To deal with different OSes and architectures, we use cloud technologies. Consequently, an image is created with all user specified packages inside a VM instantiated on-demand. Therefore, multiple users can create multiple images for different operating systems concurrently; obviously, this approach provides us with great flexibility, architecture independence, and high scalability.

We can speed-up the process of generating an image by not starting from scratch but by using an image already stored in the repository. We have tagged such candidate images in the repository as base images. Consequently, modifications include installation or update of the packages that the user requires. Our design can utilize either VMs or a physical machine to chroot into the image to conduct this step.

Advanced features of our design include the automatic upgrade or update of images stored in the repository. The old image can be deleted after the user verifies the validity of the new image.

Image Repository. The image repository [DvLW+11] catalogs and stores images in a unified repository. It offers a common interface for distinguishing image types for different IaaS frameworks but also bare-metal images. This allows us to include a diverse set of images contributed not only by the FG development team but also by the user community that generates such images and wishes to share them. The images are augmented with information about the software stack installed on them including versions, libraries, and available services. This information is maintained in the catalog and can be searched by users and/or other FG services. Users looking for a specific image can discover available images fitting their needs using the catalog interface. In addition, users can also upload customized images, share them among other users, and dynamically provision them. Through these mechanisms we expect our image repository to grow through community contributed images.

Metadata included in the repository includes information about properties of the images, the access permission by users and the usage. Access permissions allow the image owner to determine who can access this image from the repository. The simplest types of sharing include private to owner, shared with the public or shared with a set of people defined by a group/project. Usage information is available as part of the metadata to allow information about usage to be recorded. This includes how many times an image was accessed and by whom.

Image Registration. Once the image has been created and stored into the repository, we need to register it into the targeted infrastructure before we can instantiate it. Users requirements are simply the image, the targeted infrastructure and the kernel. The kernel is an optional requirement that allows advance users to select the most appropriate kernel for their experiments. This tool provides a list of available kernels organized by infrastructure. Nevertheless, users may request support for other kernels like one customized by them. Registering an image also includes the process of adapting it for the infrastruc-

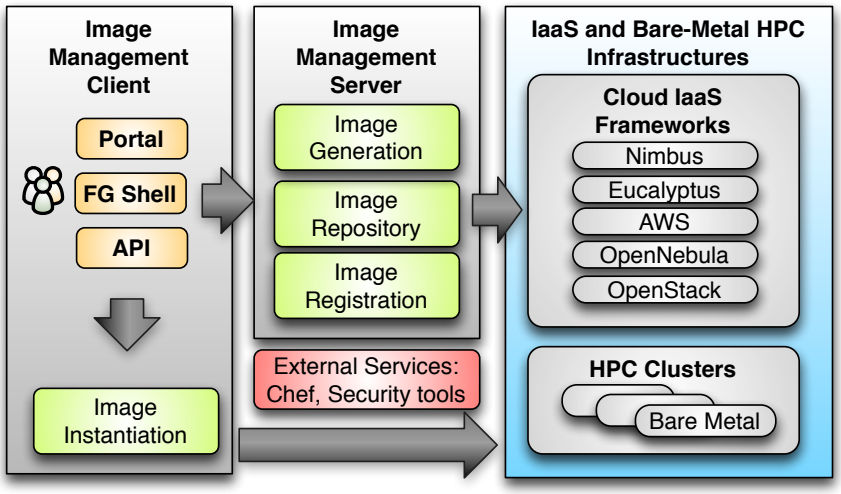

FIGURE 1.5: Architecture of Rain to conduct image provisioning on IaaS and bare-metal

ture. Often we find differences between them requiring us to provide further customizations, security check, the upload of the image to the infrastructure repository, and registering it. These customizations include the configuration of network IP, DNS, file system table, and kernel modules. Additional configuration is performed depending on the targeted deployed infrastructure.

In the HPC infrastructure the images are converted to network bootable images to be provisioned on bare-metal machines. Here, the customization process configures the image, so it can be integrated into the pool of deployable images accessible by the scheduler. In our case this is Moab. Hence, if such an image is specified as part of the job description the scheduler will conduct the provisioning of the image for us. These images are stateless and the system is restored by reverting to a default OS once the running job requiring a customized image is completed.

Images targeted for cloud infrastructures need to be converted into VM disks. These images also need some additional configuration to enable VM's contextualization in the selected cloud. Our plan is to support the main IaaS clouds, namely Eucalyptus, Nimbus, OpenStack, OpenNebula, and Amazon Web Service (AWS). As our tool is extensible, we can also support other cloud frameworks.

Of importance is a security check for images to be registered in the infrastructures. A separate process identifies approved images, which are allowed to be instantiated in FutureGrid. Approval can be achieved either by review or the invocation of tools minimizing and identifying security risks at runtime. Users may need to modify an image to install additional software not available dur-

ing the image generation process or to configure additional services. Modified images need to go through some additional tests before they can be registered in the infrastructure. To perform these security tests, we plan to create a high level platform for instantiating the images in a controlled environment such as a VM with limited network access. Hence, we can perform some tests to verify the integrity of the image, detect vulnerabilities and possible malicious software

If the image passes all the tests, it is tagged as approved. To provide authentication and authorization images may interface with the FG account management. The process of registering an image only needs to be done once per infrastructure. Therefore, after registering an image in a particular infrastructure, it can be used anytime to instantiate as many VMs or in case of HPC as many physical machines as available to meet the users requirements.

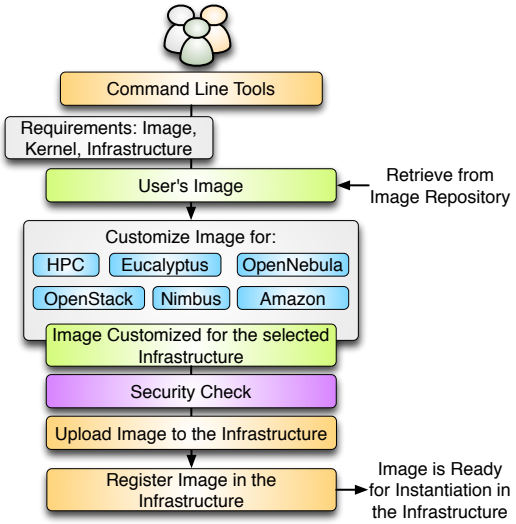

FIGURE 1.6: Image Registration

Dynamic Provisioning with RAIN Due to the variety of services and limited resources accessible in FG, it is necessary to enable a mechanism to provision needed services onto available resources and suspend or shutdown services that are not utilized. This includes the assignment of resources to different IaaS and PaaS frameworks. We we need to develop a convenient abstraction that allows us to hide the many underlying tools and services to accomplish this task. We observed the following needs from the FutureGrid user community:

1. In contrast to Grid frameworks such as Globus, we are not only interested in interfacing to the job management system or file transfer.

2. In contrast to IaaS environments, we are not just interested in provisioning an image as part of the virtualized environment. Instead, we would like to be able to "provision" the entire IaaS framework.

3. In contrast to environments based on a single operating system, we would like to manage the deployment on multiple base OS including the choice of which virtualization technology is used.

4. In contrast to just focusing on virtualized environments, we would also like to enable an environment allowing performance comparisons between the virtualized and the non-virtualized versions of applications, e.g. comparing HPC, vs. IaaS frameworks. Hence, it is important to recognize that this comprehensive view of "raining" an environment is a significant contribution of FG and allows comparative studies that are otherwise not easily possible.

We have developed, as first step to address this challenge, a sophisticated image management toolkit that allows us to not only provision virtual machines, but also provision directly onto bare-metal [DvLWF12]. Thus, we use the term *raining* to indicate that we can place arbitrary software stack onto a resource. The toolkit to do so is called *rain*.
Rain will make it possible to compare the benefits of IaaS, PaaS performance issues, as well as evaluating which applications can benefit from such environments and how they must be efficiently configured. As part of this process, we allow the generation of abstract images and universal image registration with the various infrastructures including Nimbus, Eucalyptus, OpenNebula, OpenStack, but also bare-metal via the HPC services. It is one of the unique features about FutureGrid to provide an essential component to make comparisons between the different infrastructures more easily possible [vLDWF12]. Our toolkit rain is tasked with simplifying the creation and deployment of customized environments.
Internally rain may use a multitude of tools and components suitable to conduct the task indicated by the command line tool. This may include Moab, xCAT, TakTuk [tak12], and even IaaS frameworks where appropriate. It is important to recognize that, in order to allow repeatable experiments, the rain command will have to interact with a multitude of services. In future, rain will allow specifically the recording of the resources participating in the experiment. Eventually, the experiment can be shared with other users and replicated.
As a result, rain is a pivotal process in making the FG deployment unique and applicable to any set of scientific researchers requiring rapid deployment IaaS, PaaS, and HPC environments (see Figure 1.7). Thus, rain will offer the following main features:

- Create customized environments on demand.
- Compare different infrastructures.

- Move resources from one infrastructure to another by changing the image they are running plus doing needed changes in the framework.
- Ease the system administrator burden for creating deployable images.
- Access to repeatable experiments.

Examples for "raining" are the deployment of an Hadoop cluster to run an experiment, the instantiation of a set of VMs with an specific OS, or the deployment of a virtual cluster based on SLURM [**?**, wwwi]. We envision that commands such as

```
fg-rain --hadoop -x india -m 10 -j jobscript.sh
fg-rain -os ubuntu11.04 -s sierra -m 25 -I
fg-rain --cluster slurm -x sierra -m 34
```

will be transparent enough for users to achieve provisioning on both bare metal and virtualized infrastructures (cloud). It is obvious that such a command will be extremely powerful and provide an important mechanism for abstracting the many different tools and services that are needed to accomplish the task. In this way, users don't need to be aware of the underlining details of each infrastructure. Hence, the command rain will provide the high level interface to the FG fabric, which is essential to create deployment workflows in a simple fashion.

1.3.3.2 Monitoring, Information, and Performance Services

A critical component of FutureGrid is the ability to monitor the expected behavior of its systems and services. This is especially important due to the experimental nature of the FutureGrid mission. Monitoring activities in FutureGrid include testing the functionality and performance of FutureGrid services using Inca, collecting usage data with Netlogger, cluster monitoring with Ganglia, and network monitoring with perfSONAR and SNAPP. Administrators and users can then easily access the monitoring data generated by each tool through the user portal. Programmatic access to the monitoring data is also available through each of the tools and work is ongoing to integrate the monitoring data into a common messaging system using AMQP for easier access. Each monitoring tool is described further in the subsections below. We also provide performance tools, PAPI and Vampir, into the image generation process for both virtual and bare-metal images.

Inca. Inca [SEHO07] is a monitoring framework designed to detect cyber-infrastructure problems by executing periodic, automated, user-level monitoring of CI software and services. Inca currently executes over 200 tests for FutureGrid's Cloud, Grid, HPC, and internal services: Eucalyptus, Ganglia, GCC, Genesis II, Globus GRAM, Globus GridFTP, HostCert/CRL, HPCC, IMPI, Inca, Infiniband (Sierra), JIRA, LDAP, Modules, MongoDB, myHadoop, Openstack, NetLogger, Nimbus, OpenMPI, PAPI, perfSONAR,

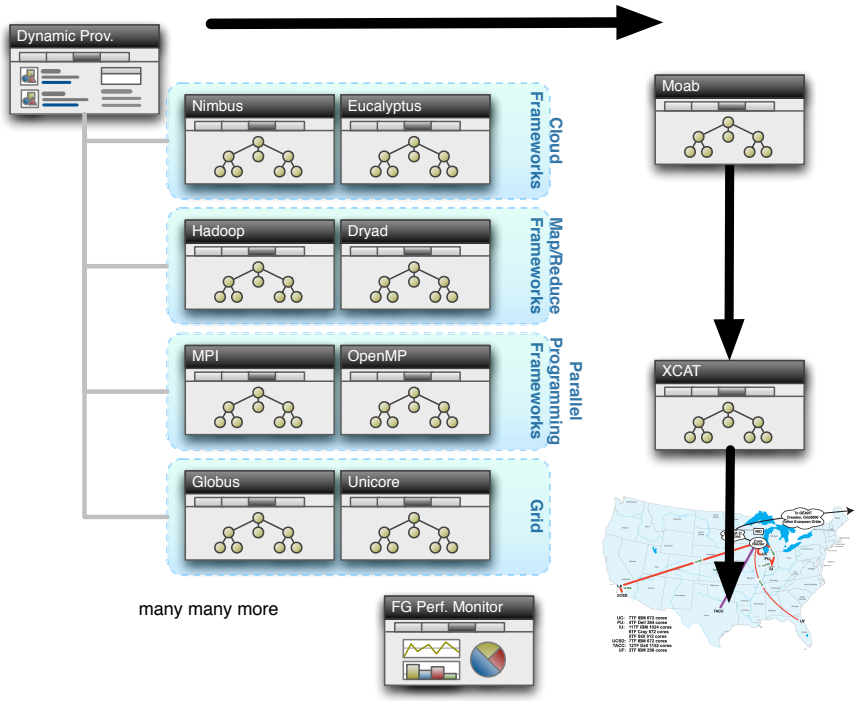

FIGURE 1.7: The concept of raining allows to dynamically provision images on bare-metal, but also in virtualized environments. This allows performance experiments between virtualized and non virtualized infrastructures.

PGCC, the FG Portal, SSH, Torque/Moab, Unicore, VampirTrace, Wiki, and XCAT. It also collects performance measurements for each of the Cloud tools and HPCC performance data from the HPC partitions.

Netlogger. Netlogger [TJC+98] is a tool for debugging and analyzing the performance of complex distributed applications. It is currently being leveraged to collect privileged usage data (number of deploy VMs and number of unique users) using small probes that execute with administrator credentials every hour for each of the Cloud tools: Eucalyptus, Nimbus, and Openstack.

Ganglia. Ganglia [MCC04] is a cluster monitoring tool used to monitor each of the FutureGrid clusters. It collects CPU, memory, disk, and network usage statistics for each cluster node. The monitoring data is then aggregated and published at a centralized Ganglia server.

perfSONAR. perfSONAR [HBB+05] is an infrastructure for network performance monitoring. FutureGrid has deployed the perfSONAR-BUOY infras-

tructure to perform a full mesh of BWCTL measurements (regularly-scheduled Iperf measurements). Currently, we have a basic setup of sixty second 1G TCP measurements that are run every two hours.
SNAPP SNAPP [SNA07] is a network statistical data collection and visualization tool that is used to collect high-performance, high-resolution SNMP data from the FutureGrid network.

1.3.4 Experiment and Project Management

Users obtain access to FutureGrid by submitting projects describing the nature and merit of their activity. Currently most projects tax FutureGrid not in number of resources requested but in the nature and support of requested software and these issues are used in evaluation. We do not approve projects such as production science that are more suitable for other XSEDE sites. There are currently no restrictions on nationality or type (academic, government, industry) of users. We require however that the use of FutureGrid be documented and results be shared with the community. The FutureGrid portal https://portal.futuregrid.org/ [wwwb] has an open list of all projects, their results, papers on its development and reports on its progress.
FutureGrid allows both federated and non federated experiments. Due to the variety of technologies that are supported not all of them integrate yet well into a completely federated environment. We base our federated infrastructure on LDAP where possible, while using public keys. Surprisingly, it was relatively easy so far to provide an IdP (Identity Provider) based on a simple verification process that included a google search on academic publications, participation in source code development of established projects, or lookup on university Web sites. We also identified that we have users that are and probably never will be part of the US campus bridging IdP based on InCommon. An OpenID can be used to be associated with the portal account, effectively providing SSO with services such as Googledocs. This is especially of importance as we believe that in order to support the long tail of science we need to interface with such tools as they provide via google scholar, and collaborative document preparation popular in the education community.
Experiments are currently done as part of projects that are vetted and approved. Once a project is approved users can conduct experiments. A result of such experiments can be a template that can be used to reproduce such an experiment or even an image that others can use by reusing the software stack promoted by this image.
We are in the process of developing a number of tools that together provide a sophisticated experiment management environment while leveraging lessons learned from our earlier experiences [vLYH+09, vL06]. This is done as part of a targeted software architecture that we make public as part of our activities. This architecture reuses a number of tools including Nimbus, OpenStack, OpenNebula, Eucalyptus, Globus, Unicore, Genesis II, Pegasus [CD11, VDRB11]. For dynamic provisioning on bare metal and VMs

we are using the term *rain* and developed the fg-rain prototype that not only places the OS on the resources (virtualized and non-virtualized) but also assembles the OS and the software stack as part of an image generation process[vLFW+10, DvLW+11, DvLWF12].

1.3.4.1 Interactive Experiment Management

The design philosophy of this set of tools is a Unix-style one where a user can use a set of independent tools together (via the command line or a script) to accomplish a complex goal.

Interactive experiment management is currently an ongoing research activity within FutureGrid. Various approaches to this exist within FutureGrid. One of these approaches is to create an experiment harness with existing and new tools such as TakTuk and the FutureGrid Host List Manager. These tools are used in conjunction with other tools such as Torque and Nimbus provisioning commands and even the Unix script command. TakTuk is a cluster fork or parallel shell type tool developed as part of the Grid 5000 project. The TakTuk user interface is a command line program that allows a user to easily and efficiently execute commands on sets of remote computer systems and transmit the output of these commands back to the TakTuk program. TACC developed the Host List Manager - a set of command line programs that discovers what FutureGrid resources have been provisioned to a user (via Torque and Nimbus), organizes resources into groups (based on the tag(s) that the user assigns to each resource), and generates host list files for use by TakTuk or similar parallel shell tools.

A FutureGrid user can therefore use Torque and Nimbus commands to provision resources, use the Host List Manager to organize those resources into groups, execute commands on the resources via TakTuk, and record their entire session via the Unix script program.

TakTuk and the Host List Manager have been deployed on the Alamo, Hotel, India, and Sierra FutureGrid clusters and are available in production to FutureGrid users. The deployment includes the software itself and modules for including the software into a user's environment.

1.3.4.2 Workflow-Based Experiment Management

Scientific workflows have been popular within the Grid community to coordinate the execution of large scale workflows utilizing resources from a variety of Grids. It is obvious that part of this strategy can be reused within the Cloud environment. This strategy is followed by the Pegasus project that provides to FutureGrid the promise to allow workflow based experiment Management to the FutureGrid community.

1.3.4.3 DevOps-Based Experiment Management

In addition to the experiment management strategies listed above, we also support experiment management with tools that recently have become synonymous with DevOps. Here templates, scripts, or recipies are used to formulate in a templated fashion deployment descriptions to coordinate the creation of an experiment environment that includes resources as part of the compute fabric, the operating system and/or virtuualization environments, and the instantiation of high-level platforms as part of the overall deployed experimental services. Such a mechanism is possible while leverageing tools such as Puppet or chef as well as integrating them as part of our RAIN toolkit. The goal is to create small test deployments that with little effort can be rained onto larger resource fabrics when desired. It also allows the total reconfiguration of the environment while proposing the model of "nothing installed on the machine". Hence such an experiment management framework could also be used to bootstrap the previous two Management frameworks that are positioned at a higher level and require certain software to be readily be available.

1.3.4.4 Rain-Based Experiment Management

As one of the key components and services of FutureGrid is to dynamically provision operating systems, IaaS, and PaaS frameworks onto different resources, as well as the dynamic allocation based on user demand, it will be possible to create and recreate environments and experiment resources based on predifined specifications. This will help in setting up suitable experiment frameworks that allow comparision of features across infrastructures, platforms, operating systems, and other high level tools. Hence it builds an elementary basic unit that can be used in workflow-based, interactive, as well as DevOps-based experiments. Without doubt one of the most sophisticated services planed and offered in FutureGrid. To facilitate ease of use we take the ideas presented in [vLYH+09] and expose the funtionlity throug a convenient comand shell while also working towards integraton into a portal.

1.3.5 Operations Services

Authentication and Authorization. We have devised our authentication strategy on LDAP and enabled the management of users through the portal. Our account policy is based on that all users must (a) have a portal account, (b) be in an approved project, and (c) have an ssh key uploaded Accounts for Eucalyptus must currently be separately applied for as the version of Eucalyptus we have access to is not yet integrated with LDAP. The LDAP servers are distributed and also Nimbus is already integrated with our LDAP server. LDAP provided an excellent mechanism for our security needs. All clouds Nimbus, Eucalyptus, OpenStack, OpenNebula we consider for deployment are currently or will support LDAP. We consulted with security experts that gave the recommendation to use LDAP. Originally we wanted to interface also with

InCommon, for account vetting, but found that vetting accounts so far was not that much work and that the attributes provided by InCommon does at this time not allow us to eliminate our account vetting activities as confirmed by experts from the InCommon community.

Cloud Accounting. An important part for FG is to be able to present accounting information for our deployed cloud environments. This has been surprisingly more challenging as we originally anticipated. First, the cloud frameworks support for accounting is minimal or non-existing. In addition, we are also investigating mechanisms to not only parse and monitor Eucalyptus logs, but also to identify that we can integrate that information into a system such as Gold. However, the developers of Gold have recently discontinued support for it and recommend instead a commercial solution. We will reevaluate our strategy as it was based on Gold.

HPC Accounting. Accounting on HPC systems is important to identify how the systems were used and utilized. Although integration with the XSEDE central repository would be possible, we identified together with University of Buffalo several challenges. First, the metrics we are interested in such as VM monitoring and utilization are not in the database and require a significant modification. Second due to the reconfigurable nature of FutureGrid the number of nodes associated to the HPC services is not static and does not fit the current analysis and recoding model. Third, the type of jobs we ran is not as typical and CPU bound. For example we find many users testing scientific workflows instead of conducting large scale number crunching on our resources.

Intelligent Resource Adaptations. Once we have accounting information we will start an effort on utilizing the accounting information to integrate it into a dynamic provisioning mechanism as part of RAIN. E.g. if we detect that users do not want to use HPC resources or Eucalyptus resources, the machines hosting the compute nodes for these services can be assigned to for example Nimbus if Nimbus reports to us hat more resources are needed. This will lead to a metascheduler for cloud and HPC resources.

1.3.6 Development Services

Due to the size of the project it is important to provide a good infrastructure in support of our collaborative efforts. This includes the deployment of a task management system, a wiki, and a continuous integration environment.

1.3.7 Portal

FutureGrid presents a sophisticated portal based on Drupal that integrates not only with sever of our services, but also provides the ability to foster a community. Of especial importance is the integration of several workflows that make the review and the approval process of projects very simple.

1.4 Service Deployment

In Table 1.3 we list elementary services that have been deployed on which resources in FG.

TABLE 1.3: Deployed services

	India	Sierra	Hotel	Foxtrot	Alamo	Xray	Bravo	Echo
myHadoop	(d)	(d)			(d)			
Nimbus	(d)	(d)	(d)	(d)				
Eucalyptus	(d)	(d)						
ViNe	(i)	(d)	(i)	(d)	(i)			
Genesis II	(d)	(d)			(d)	(d)		
Unicore	(d)	(d)				(d)		
MPI	(d)	(d)	(d)	(d)	(d)	(d)	(d)	
OpenMP					(d)			
ScaleMP	(d)							
Ganglia	(d)		(d)					
Pegasus	(i)	(i)	(i)	(i)	(i)			
Inca	(d)	(d)	(d)	(d)	(d)	(d)		
Portal	(i)	(i)	(i)	(i)	(i)	(i)		
PAPI					(d)			
Vampir	(d)					(a)		
Vampir Trace	(d)					(d)		
RAIN	(d)							

Legend:
(d) deployed (a) can be made available upon request (i) information available in the portal

1.5 Applications using FutureGrid

More than 220 projects and 920 users are registered in FutureGrid. These projects cover a wide range from application to technology and from research to education. Recent projects have focused on integration testing for XSEDE, image management and dynamic provisioning on bare metal [DvLWF12], scalability test for cloud provisioning [vLDWF12]. These projects are ground braking as they introduce a testbed environment for XSEDE an also allow user facing dynamic provisioning, something that is not normally offered by other resources. The scalability experiment showed certain limitations with standard cloud setups for use cases typical for scientific applications.
Additionally, we list in Table 1.4 a number of projects that have undertaken on FutureGrid to provide an overview of the wide variety of projects. Each

project must report their success and findings on the FutureGrid Web site [wwwb]. Currently, our portal is quite streamlined for the entire workflow related to user account creation, project creation, and reporting. In addition application user forums can be established for the projects if the project lead wishes. This makes it possible to have projects created in just a very short time period. In the next sections we will present some selected projects. To see all of the projects conducted on FutureGrid we recommend to visit the FutureGrid portal.

1.5.1 Privacy preserving gene read mapping using hybrid cloud

An example project investigationg security aspects of hybrid clouds is found at [Cheb]. In this project the researchers study the possibility of doing reads mapping using hybrid cloud, in order to utilize public computing resources while preserving the data privacy. The topic of this research is very timely in order to address requirements about privacy in bioinformatics as more and data are generated. The tem conducting this research also intends to increase the data processing speed in the area of bioinformatics andreplace current read mapping tools. A typical experiment on FutureGrid will run for about two to three days.

One of the most important analyses on human DNA sequences is read mapping, which aligns a large number of short DNA sequences (called reads) produced by sequencers to a reference human genome. The analysis involves intensive computation (calculating edit distances over millions upon billions of sequences) and therefore needs to be outsourced to low-cost commercial clouds. This asks for scalable privacy-preserving techniques to protect the sensitive information sequencing reads contain. Such a demand cannot be met by the existing techniques, which are either too heavyweight to sustain data-intensive computations or vulnerable to re-identification attacks. Our research, however, shows that simple solutions can be found by leveraging the special features of the mapping task, which only cares about small edit distances, and those of the cloud platform, which is designed to perform a large amount of simple, parallelizable computation. We implemented and evaluated such new techniques on a hybrid cloud platforms built on FutureGrid. In our experiments, we utilized specially-designed techniques based on the classic "seed-and-extend" method to achieve secure and scalable read mapping. The high-level design of our techniques is illustrated in Figure 1.8.

Here, the public cloud on FutureGrid is delegated the computation over encrypted read datasets, while the private cloud directly works on the data. Our idea is to let the private cloud undertake a small amount of the workload to reduce the complexity of the computation that needs to

The researchers reported that their solution method is performed on the encrypted data, while still having the public cloud shoulder the major portion of a mapping task. They constructed a hybrid environment over FutureGrid

TABLE 1.4: Selected FutureGrid Projects

Project	Institution	Details
Educational Projects		
System Programming and Cloud Computing	Fresno State	Teaches system programming and cloud computing in different computing environments
REU: Cloud Computing	University of Arkansas	Offers hands-on experience with FutureGrid tools and technologies
Workshop: A Cloud View on Computing	Indiana University	Boot camp on MapReduce for faculty and graduate students from underserved ADMI institutions
Topics on Systems: Distributed Systems	Indiana University	Covers core computer science distributed system curricula (for 60 students)
Interoperability Projects		
SAGA	Rutgers	Explores use of FutureGrid components for extensive portability and interoperability testing of Simple API for Grid and scale-up and scale-out experiments
Applications		
Metagenomics Clustering	North Texas	Analyzes metagenomic data from samples collected from patients
Genome Assembly	Indiana School of Informatics	De novo assembly of genomes and metagenomes from next generation sequencing data
Physics: Higgs boson	Virginia	Matrix Element calculations representing production and decay mechanisms for Higgs and background processes
Business Intelligence on MapReduce	Cal State LA	L.A. Market basket and customer analysis designed to execute MapReduce on Hadoop platform
Computer Science Data Transfer Throughput	Buffalo	End-to-end optimization of data transfer throughput over wide-area, high-speed networks
Elastic Computing	Colorado	Tools and technologies to create elastic computing environments using IaaS clouds that adjust to changes in demand automatically and transparently
The VIEW Project	Wayne State	Investigates Nimbus and Eucalyptus as cloud platforms for elastic workflow scheduling and resource provisioning
Technology Project		
ScaleMP for Gene Assembly	Indiana University	Pervasive Technology Institute (PTI) and Biology Investigates distributed shared memory over 16 nodes for SOAPdenovo assembly of Daphnia genomes
XSEDE	Virginia	Uses FutureGrid resources as a testbed for XSEDE software development
Cross Campus Grid	Virginia	Work on bridging infrastructure across different university campus

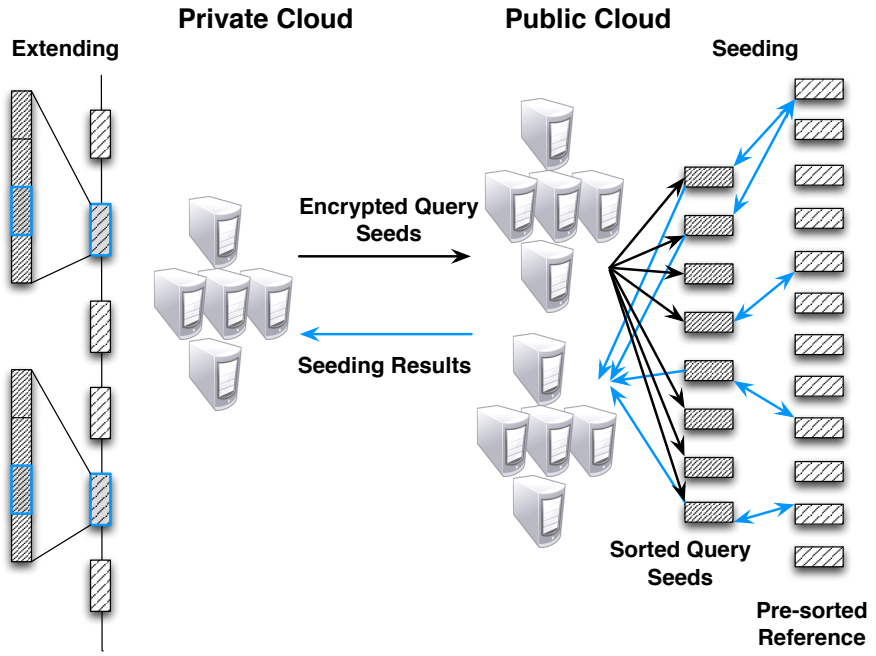

FIGURE 1.8: Using a public and private cloud for gener read mapping

while using the following two modes: First, a virtual mode in which 20 nodes of FutureGrid wer used as the public cloud and 1 node was configured as private cloud. Second, a real mode, in which nodes on FutureGrid were used as the public cloud and the computing system within the School of Informatics and Computing as the private cloud.

Our experiments demonstrate that our techniques are both secure and scalable. We successfully mapped 10 million real human microbiome reads to the largest human chromosome over this hybrid cloud. The public cloud took about 15 minutes to conduct the seeding and the private cloud spent about 20 minutes on the extension. Over 96/

1.5.2 SAGA on FutureGrid

An example for an interoperability project is the Simple API for Grid Applications (SAGA) [sag12, fg4]. It is based on an OGF standard [wwwe], and defines a high level, application-driven API for developing first-principle distributed applications, and for distributed application frameworks and tools. SAGA provides API implementations in C++ and Python, which interface to a variety of middleware backends, as well as higher level application frameworks, such as Master-Worker, MapReduce, AllPairs, and BigJob [sag]. For all

those components, we use FutureGrid and the different software environments available on FG for extensive portability and interoperability testing, but also for scale-up and scale-out experiments. These activities allow to harden the SAGA components described above, and support CS and Science experiments based on SAGA. FG has provided a persistent, production-grade experimental infrastructure with the ability to perform controlled experiments, without violating production policies and disrupting production infrastructure priorities. These attributes, coupled with FutureGrid's technical support, have resulted in the following specific advances in the short period (1) use of FG for Standards-based development and interoperability tests; (2) Use of FG for analysing and comparing programming models and run-time tools for computation and data-intensive science; (3) development of tools and frameworks; (4) cloud interoperability experiments; and (5) data intensive applications using MapReduce. SAGA will continue to use FG as a resource for SAGA development for testing of other SAGA based components, to widen the set of middleware used for testing to enhance the scope and scale of our scalability testing; and to test and harden our deployment and packaging procedures.

More details about SAGA on FutureGird, can be found on the Portal [fg4].

1.6 Optimizing MapReduce

MapReduce has been introduced by the information retrieval community, and has quickly demonstrated its usefuleness, scalability and applicability. Its adoption of data centered approach yields higher throughput for data-intensive applications.

One FutureGrid Project centers around the investigation and improvement of MapReduce. As part of the project the team identifies the inefciency of various aspects of MapReduce such as data locality, task granularity, resource utilization, and fault tolerance, and propose algorithms to mitigate the performance issues. Extensive evaluation is presented to demonstrate the effectiveness of their proposed algorithms and approaches. Observing the inability of MapReduce to utilize cross-domain grid resources motivates the proposal to extend MapReduce by introducing a Hierarchical MapReduce (HMR) framework. The framework is tested on bioinformatics data visualization pipelines containing both single-pass and iterative MapReduce jobs, a workow management system Hybrid MapReduce (HyMR) is developed and built upon Hadoop and Twister. A detailed performance evaluation of Hadoop and some storage systems, which provides useful insights to both framework and application developers is provided. The work resulted in several papers and a PhD thesis [Guo12].

1.7 Sensor Cloud

The Pervasive Technology Institute, Anabas, Inc., and Ball Aerospace have successfully collaborated to complete a cloud-based message passing middleware, referred to as the Sensor Cloud [**?**, **?**], to provide a research test bed for sensor-centric/Internet of Things application development.
The objective of the Sensor Cloud Project was to provide a general–purpose messaging system for sensor data. For our purposes a sensor is defined as anything producing a time–ordered data stream. Examples of sensors include physical devices like web cams, robots, or a Kinect motion sensing input device. However sensors can also be Twitter tweets, the results from some computational service, even a PowerPoint presentation, anything that produces a time dependent data series can be a sensor.
The components of the Sensor Cloud are the Sensor Grid Server for message routing, a sensor grid building and management tool called the Grid Builder and a robust Application API for developing new sensors and client applications. The Grid Builder has an intuitive interface for setting sensor policies as well as easy deployment and management of sensors across global networks. The key design objective of the Sensor Grid API is to create a simple integration interface for any third party application client or sensor to the Sensor Grid Server. This objective is accomplished by implementing the publish/subscribe design pattern which allows for loosely–coupled, reliable, scalable communication between distributed applications or systems.
The Sensor Cloud was developed and tested using the Future Grids Open Stack Infrastructure as a Service (IaaS) cloud. The Future Grid provided a scalable geographically distributed environment in which to measure the Sensor Cloud system performance.

1.7.1 FutureGrid as a test bed for XSEDE (eXtreme Science and Engineering Discovery Environment)

In 2008, the NSF announced a competition for the follow-on to the TeraGrid project, known as eXtreme Digital (XD). In 2009, two proposal teams, one led by NCSA and the other by SDSC, were selected to prepare full proposals. In July 2010, the proposals were delivered, and in late 2010, the NCSA-led team was awarded the project, albeit with the instruction to incorporate the best ideas and personnel from the SDSC proposal.
The NCSA-led team proposed XSEDE the eXtreme Science and Engineering Discovery Environment. The XSEDE architecture is a three-layer, federated, system-of-systems architecture. It includes the use of standard Web Services interfaces and protocols that define interactions between different components, tools, and organizations. Many different science communities will use XSEDE, whether at a national supercomputing center or through its delivery of campus

bridging among research groups around the US. XSEDE will be, upon completion, one of the critical components of the national cyberinfrastructure.
The XSEDE Web Services architecture is based on a number of interchangeable components that implement standard interfaces. These include: 1) RNS 1.1 [MGT10], for Unix-directory-like namespace (e.g. /home/sally/work); 2) OGSA-ByteIO [Mor05] for POSIX file-like operations (create, read, update, delete); 3) OGSA Basic Execution (OGSA-BES) [GNPM08] for executing and managing jobs (create_activity, get_status, delete); and 4) WS Trust Secure Token Services (STS) [OAS07] for identity federation. The architectural goals have clearly-specified, standard interfaces that anybody can implement, and use best-of-breed components.
The initial realization of the XSEDE web services architecture uses interoperable implementation from two different software stacks: UNICORE 6 [Sne03] and Genesis II [6]. Together, UNICORE 6 and Genesis II provide a rich set of capabilities in the areas of data, computation, and identity management. These capabilities are grouped into two configuration items (CIs): Execution Management Services (EMS) and the Global Federated File System (GFFS). The Execution Management Services CI is concerned with specifying, executing, and more generally managing jobs in the XSEDE grid. EMS capabilities include but are not limited to the following:

1. The ability to specify both single jobs and parameter space jobs in JSDL. Specified jobs may be sequential jobs or parallel (MPI) jobs.
2. The ability to manage jobs through their lifetime, i.e., from specification and submission to a compute resource to status checking and management during execution, as well as final cleanup.
3. A grid-queue (meta-scheduler) that matches jobs to a defined, configurable, set of execution services and load balances between them.
4. The ability to specify either a single compute resource as a target, e.g., a particular queue on Ranger, or to specify a global metascheduler/queue as the target and have the metascheduler select the execution endpoint.
5. The ability to add compute resources (e.g., queues on specific machines such as Ranger, Alamo, Kraken, or local campus queues such as Centurion at UVA) into the XSEDE namespace and subsequently target jobs at them.
6. The ability to create meta-schedulers/queues and configure them to use (schedule on) different compute resources.
7. A command line interface (CLI) to interact with grid compute resources.
8. A graphical user interface (GUI) to interact with and manage the backend grid compute resources. This includes, but is not limited to tools to: create and execute job descriptions, manage grid-queues, and manage access to resources.

9. A set of Java classes (and associated APIs) to interact with and manage the backend grid resources.

The Global Federated File System presents a file-system-like view of diverse resources types located at service providers, campuses, research labs, and other institutions. Resources (e.g., files, directories, job submission queues) are mapped into a single global path-based namespace. Resources can be accessed by their path name in a location-, replication-, migration-, and failure-transparent manner. Resources can be accessed via command line tools (a grid shell), a graphical user interface, or via the users local file system and a FUSE mount.
The GFFS provides a number of capabilities. These capabilities include but are not limited to the following:

1. A single, secure, shared global namespace for a diversity of resource types. For example, a single namespace can include files, directories, execution services, execution queues, secure token services, and executing jobs.

2. A three-level naming scheme consisting of location-independent, human-readable names (paths) that map globally unique resources identities – identities that in turn can be mapped (bound) to one or more resource instances. Collectively, the three layers provide an easy-to-use name space that transparently handles heterogeneous configurations for location, failure, replication, migration, and implementation.

3. The ability to securely map (share) Service Provider (SP), local, lab, and campus data into the shared global namespace.

4. The ability to securely map (share) SP, local, lab, and campus compute resources into the global namespace.

5. The ability to securely map (share) SP, local, lab, and campus identity resources into the global namespace.

6. The ability to transparently access from both campuses and national supercomputing centers to the global shared namespace, via either the file system (e.g., FUSE) or command line tools and libraries. Such access includes the ability to perform create, read, update, and delete operations on files, directories, and other resource types.

7. A command line interface (CLI) to interact with backend grid resources. In particular, this CLI allows interaction with Open Grid Forum RNS, ByteIO, WS-Naming, and BES services, as well as WC3 WS-Trust Secure Token Services.

8. A graphical user interface (GUI) to interact with and manage the backend grid resources.

9. A set of Java classes (and associated APIs) to interact with and manage the backend grid resources.

10. The ability to integrate with existing, legacy, XSEDE Kerberos and MyProxy [BYBS03] authentication mechanisms.

Before the capabilities of the CIs can be delivered to end users, they must be tested in as realistic an environment as possible. The testing needs to reflect both production workloads as well as extreme conditions including those that might crash production machines. What is therefore needed is an infrastructure that closely mimics the intended execution environment. This environment would be: 1) a collection of geographically separated parallel processing systems of various types connected by high speed networks; 2) connected to the national network backplane; 3) able to be used for testing without negatively impacting the production workloads at the national centers. FutureGrid fits these requirements perfectly.
In December 2011, the XSEDE-testgrid (XTG) was brought up on FutureGrid and University of Virginia resources as part of the first integrated test plan for execution management services (EMS) and the Global Federated File System (GFFS). The root of the RNS namespace was set up at Virginia and UNICORE 6 servers brought up on X-Ray at Indiana (the Cray that is a small version of Kraken) and Sierra at SDSC. Genesis II servers were brought up at Virginia and TACC. Genesis II clients were brought up at Indiana, Virginia, TACC, and PSC on Blacklight.
Over the next two months, the tests described in the EMS and GFFS test plans were then executed by the XSEDE Software Development and Integration (SD&I) team. This led to the discovery of several minor problems as well as a number of desired feature enhancements. Minor bugs and integration issues were resolved and the increment 1.0 of EMS and GFFS were completed and, in March 2012, they were turned over to the XSEDE operations team for learning and testing purposes.
XSEDE testing activities on FutureGrid bifurcated in April 2012. The XSEDE operations team began doing its own experimentation and testing on FutureGrid resources, and SD&I continued to use the XTG as a test infrastructure in particular working with the XSEDE Campus Bridging team on the Campus Bridging Pilot project. By the end of May 2012, operations had completed its EMS testing and was ready to start GFFS testing. In late June of 2012, SD&I began testing the second release increment of the EMS and GFFS CIs.
As part of the campus bridging pilot project, data resources at three universities were incorporated into the XSEDE test grid in June 2012: Indiana University, Louisiana Tech, and LSU. These resources will be used to perform typical campus bridging test cases and to capture issues that arise with real users in the wild. To support job execution on a wider set of resources than are configured in the XTG, the compute resources in the Cross Campus Grid (XCG) were linked into the XTG.
The XCG is a Genesis II-based grid run by the University of Virginia with

resources on FutureGrid and at the University of Virginia. The XCG has been in production use for over three years, during which time it has executed over 1.3 million jobs for a number applications in areas such as economics, materials science, systems engineering, biology, and physics. Many of the XCG jobs have run on FutureGrid resources via the XCG.

FutureGrid has been an invaluable resource for XSEDE in testing the new generation of standards-based software described in the XSEDE proposal. Without FutureGrid, the XSEDE SD&I team would have been forced to use significantly smaller test machines at the centers and to execute its tests alongside production applications on the network and local parallel file systems. Instead, thanks to FutureGrid, the XSEDE team has been able to take advantage of a risk-free testing environment, collaborative systems administrators, and the similarity between the FutureGrid resources and its own. Given its ideal qualifications, we expect XSEDE to continue to use FutureGrid as a test environment.

1.8 Educational Outreach

As described earlier, the hardware resources and middleware of FutureGrid enable the creation of user-defined, dynamically-provisioned environments and support a "viral" user contribution model. Education and outreach in FutureGrid leverage its underlying technology to deliver a system that enables advanced Cyberlearning (learning and teaching enhanced or enabled by cyberinfrastructure) and to lead initiatives to broaden participation by lowering barriers to entry to the use of complex cyberinfrastructures.

In the context of education and outreach, FutureGrid technologies can aggregate and deliver user-provided educational materials including the executable software environment, middleware and applications where repeatable hands-on experiments can be conducted. Thus, at the core of FutureGrid's educational mission is the ability to create consistent, controlled and repeatable educational environments in areas of computer and computational science related to parallel, largescale or distributed computing and networking, as well as the availability, repeatability, and open sharing of electronic educational materials. FutureGrid has deployed a distributed platform where educators and students can create and access such customized environments for hands-on education and outreach activities on cloud, HPC and grid computing.

Key components of the FutureGrid cyberlearning infrastructure are plug-and-play virtual appliances and virtual networks, which are used to support hands-on educational modules that run on self-contained virtual clusters. The target audience here is prospective modal users, hence the focus is on usability and low barrier to entry. There are three central principles in this approach: (1) allowing users and groups of users to create and manage their own groups

through a Web-based portal, (2) automatically mapping these user relationships to network connections among computer resources using self-configuring Virtual Private Network (VPN) technologies; and (3) packaging of these environments in self-contained, self-configuring virtual appliances that make it possible to effectively bring together a comprehensive suite of software and middleware tailored to target activities (e.g. to study parallel processing in the context of Condor, MPI, or Hadoop) while hiding the system's underlying complexity from its end users.

In terms of design and implementation, educational virtual clusters on FutureGrid can be created by end users, and leverage a pre-configured Grid appliance [WF11] image, which encapsulates software environments commonly used. These appliances are deployed on FutureGrid using IaaS middleware (Nimbus, Eucalyptus, or Openstack). Once appliances are deployed, a group virtual private network (GroupVPN DecentralVPN) is self-configured to provide a seamless IP-layer connectivity among members of a group and support unmodified middleware and applications. Within a appliances, Condor is used as the core underlying scheduler to dispatch tasks. In our system, these tasks could be jobs students schedule directly with Condor, as well as tasks, which are used to bootstrap MPI , Hadoop, or Twister pools on demand. The GroupVPN virtual network and the Condor middleware are both self-configured by using a peer-to-peer Distributed Hash Table (DHT) as an information system to publish and query information about available job scheduler(s) and assign virtual IP addresses. Users create on-demand MPI, Hadoop, and Twister virtual clusters by submitting Condor jobs, which are "wrappers" for dispatching and configuring the respective run-time systems. MPI tasks are submitted together with the job that creates an MPI ring, while Hadoop and Twister tasks are submitted to the virtual cluster using standard tools, respectively. The entire system can be seamlessly deployed on a managed IaaS infrastructure, but it is also easily installable on end-user resources the same virtual appliance image is used in both environments. On a managed cloud infrastructure, IaaS middleware is used to deploy appliances, while in desktop/user environments, appliances are deployed with the native user interface of a virtual machine monitor. The appliance has been tested on widely-used open-source and commercial desktop and server virtualization technologies; the same image can be instantiated on VMware, KVM, and VirtualBox on x86-based Windows, MacOS and Linux systems.

FutureGrid has been used successfully in hands-on activities in semester classes at universities, week-long workshops and summer schools, as well as in short tutorials at conferences. Such educational and outreach activities have included:

- "A Cloudy View on Computing": A hands-on workshop for faculty from historically black colleges and universities (HBCUs) conducted on June 2011 at Elizabeth City State University. The emphasis of the work was on MapReduce concepts, and hands-on exercises using two different MapReduce platforms. Lectures focused on specific case studies of

MapReduce, and the workshop concluded with a programming exercise (PageRank or All-Pairs problem) to ensure faculty members have a substantial knowledge of MapReduce concepts and the Twister/Hadoop API. The workshop highlighted two specific educational MapReduce virtual appliances: Hadoop and Twister.

- Computer science courses: FutureGrid was used in courses including Distributed Systems and Cloud Computing for Data-Intensive Sciences at Indiana University, Cloud Computing courses at the University of Florida and at the University of Piemonte Orientale in Italy, and Scientific Computing at Louisiana State University. FutureGrid was used to build prototype systems and allow students to acquire in-depth understanding of essential issues in practice, such as scalability, performance, availability, security, energy-efficiency, and workload balancing. Students took advantage of FutureGrids dynamic provisioning infrastructure to switch between bare metal and virtual machine environments.

- Hands-on tutorials: The tutorial "Using and Building Infrastructure Clouds for Science" was given at the SuperComputing11 conference The tutorial had around 75 attendees, which were able to get accounts on FutureGrid and interact with the infrastructure with hands-on activities that included the deployment of their own virtual machines on the infrastructure. In addition to synchronous tutorials, the FutureGrid portal hosts several self-learning tutorials that guide its users through the core steps needed to configure, instantiate and access dynamically-provisioned educational environments.

1.9 Operational Lessons. We have already learnt several unexpected lessons.

Technology Breakdown. One highlight is the breakdown of usage with the over 100 FutureGrid projects divided 47% computer science, 27% technology evaluation, 18% life science applications, 13% other applications, 8% Education, and a small but important 3% Interoperability (some projects covered multiple categories, thus the totalis greater than 100%) [vLDWF12].
Education is actually more important and successful than the fraction indicates as a single class project implies 20-50 users of FutureGrid. Our usage profile is very different from other US national infrastructures. We also found that the diverse needs of users require significant user support but that many users did not need huge numbers of nodes. Thus we changed plans and targeted more funds at user support and less on hardware expansion. The ability to request both bare metal and virtualized nodes was important in many projects.

This was perhaps not unexpected but it is different from traditional environments with fixed software stacks. Further cloud technologies are rapidly changing on a 3-6 month cycle and in fact maturing but this requires a substantial effort from both software and systems groups to track, deploy and support. These groups must collaborate closely and for example automatizing of processes documented by the systems team through software development is helpful in providing a scalable service.

Interoperability Experiments. We saw a interrest by the community in interoperability experiments x1and infrastructure and have thus deployed endpoints for Unicore and Genesis II. In addition collaborations take place with the Grid5000 [imp12b], the SAGA project [sag12], and the OCCI project [wwwd] in the Open Grid Forum (OGF) [wwwe].

Acknowledgement

We like to thank the following people for their contributions to this chapter: Fugang Wang, Shava Smallen, Ryan Hartman, Koji Tanaka, Sharif Islam, David Hancock, Tom Johnson, John Bresnahan, Piotr Luszczek, Terry Moore, Mauricio Tsugawa, Thomas William, Andrew Younge, and the rest of the FutureGrid team.

Bibliography

[AA06] Keith Adams and Ole Agesen. A comparison of software and hardware techniques for x86 virtualization. In *Proc. of the 12th Intl Conf. on Architectural support for programming languages and operating system*, 2006.

[AFG+09] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. Above the clouds: A berkeley view of cloud computing. Technical report, University of California at Berkeley, February 2009. Available from: `http://berkeleyclouds.blogspot.com/2009/02/above-clouds-released.html`.

[Amaa] Amazon. Elastic Compute Cloud. Available from: `http://aws.amazon.com/ec2/`.

[Amab] Amazon. Simple Storage Services. Available from: `http://aws.amazon.com/s3/`.

[BFLK10] J. Bresnahan, T. Freeman, D. LaBissoniere, and K. Keahey. Cumulus : Open source storage cloud for science. In *SC2010*, 2010.

[BFLK11] J. Bresnahan, T. Freeman, D. LaBissoniere, and K. Keahey. Managing appliance launches in infrastructure clouds. In *TeraGrid 2011*, 2011.

[BYBS03] J. Basney, W. Yurcik, R. Bonilla, and A. Slagell. The credential wallet: A classification of credential repositories highlighting myproxy. In *31st Research Conference on Communication, Information and Internet Policy (TPRC 2003)*, 2003. Available from: `http://www.ncsa.illinois.edu/~jbasney/credentialwalletTPRC.pdf`.

[CB09] N.M. Mosharaf Kabir Chowdhurya and Raouf Boutaba. A survey of network virtualization. In *Computer Networks*, 2009.

[cC05] Susanta Nanda Tzi cker Chiueh. A Survey on Virtualization Technologies. Technical Report TR179, Department of Computer Science,State University of New York, 2005.

[CD11] Weiwei Chen and Ewa Deelman. Partitioning and scheduling workflows across multiple sites with storage constraints. In *9th International Conference on Parallel Processing and Applied Mathmatics*, Torun, Poland, 09/2011 2011. Available from: `http://pegasus.isi.edu/publications/2011/WChen-Partitioning_and_Scheduling.pdf`.

[CHEa] CompreHensive collaborativE Framework (Chef). Available from: `www.chefproject.org/`.

[Cheb] Yangyi Chen. Privacy preserving gene read mapping using hybrid cloud. FutureGrid Project. Available from: `https://portal.futuregrid.org/project/18`.

[Com12] Adaptive Computing. MOAB Cluster Suit Webpage. Webpage, Last access Feb. 2012. Available from: `http://www.clusterresources.com/products/moab-cluster-suite.php`.

[con] Condor: High Throughput Computing. Available from: `http://www.cs.wisc.edu/condor/`.

[DG08] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, 2008.

[DvLW+11] Javier Diaz, Gregor von Laszewski, Fugang Wang, Andrew J Younge, and Geoffrey 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)*, pages 560–564, Athens, Greece, 12/2011 2011.

[DvLWF12] Javier Diaz, Gregor von Laszewski, Fugang Wang, and Geoffrey C. Fox. Abstract image management and universal image registration for cloud and hpc infrastructures. In *IEEE Cloud 2012*, Honolulu, 2012.

[DYvL+11] Javier Diaz, Andrew J. Younge, Gregor von Laszewski, Fugang Wang, and Geoffrey C. Fox. Grappling cloud infrastructure services with a generic image repository. In *Proceedings of Cloud Computing and Its Applications (CCA 2011)*, Argone National Laboratory, Mar 2011.

[fg4] SAGA Simple API for Grid Application, FutureGrid Project. FutureGrid Project. Available from: `https://portal.futuregrid.org/project/42`.

[Fie00] Roy Thomas Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California, Irvine, 2000.

[FK99] I. Foster and C. Kesselman. Globus: a toolkit-based Grid architecture. *The Grid: Blueprint for a New Computing Infrastructure*, 1999.

[glo12] Globus Provision, 2012. Available from: `http://globus.org/provision/index.html`.

[GNPM08] A. Grimshaw, S. Newhouse, D. Pulsipher, and M. Morgan. Gfd108: Ogsa basic execution service. Web, 2008. Available from: `http://www.ogf.org/documents/GFD.108.pdf`.

[Guo12] Zhenhua Guo. *High Performance Integration of Data Parallel File Systems and Computing: Optimizing MapReduce*. PhD thesis, Indiana University, June 2012.

[HBB+05] Andreas Hanemann, Jeff W. Boote, Ericl. Boyd, Jrme Dur, Loukik Kudarimoti, Roman apacz, D. Martin Swany, Szymon Trocha, and Jason Zurawski. Perfsonar: A service oriented architecture for multi-domain network monitoring. In *In Proceedings of the Third International Conference on Service Oriented Computing (ICSOC 2005). ACM Sigsoft and Sigweb*, 2005.

[imp12a] Genesis II. Standards-Based Grid Computing, 2012. Available from: `http://genesis2.virginia.edu/wiki/`.

[imp12b] Grid'5000, 2012. Available from: `http://www.grid5000.fr`.

[imp12c] The Network Impairments device is Spirent XGEM, 2012. Available from: `http://www.spirent.com/Solutions-Directory/ImpairmentsGEM.aspx?oldtab=0&oldpg0=2`.

[jun12] The FG Router/Switch is a Juniper EX8208, 2012. Available from: `http://www.juniper.net/us/en/products-services/switching/ex-series/ex8200/`.

[KF08] K. Keahey and T. Freeman. Contextualization: Providing one-click virtual clusters. In *eScience 2008*, Indianapolis, IN, 2008.

[KFF+05] K. Keahey, I. Foster, T. Freeman, , and X. Zhan. Virtual workspaces: Achieving quality of service and quality of life in the grid. *Scientific Programming Journal*, 2005.

[KH11] Geoffrey C. Fox Kai Hwang, Jack Dongarra. *Distributed and Cloud Computing: From Parallel Processing to the Internet of Things*. Morgan Kaufmann Publishers, 2011.

[KTB11] Sriram Krishnan, Mahidhar Tatineni, and Chaitanya Baru. myhadoop - hadoop-on-demand on traditional hpc resources. Technical report, 2011.

[KV04] Vassiliki Koutsonikola and Athena Vakali. Ldap: Framework, practices, and trends. In *IEEE Internet Computing*, 2004.

[LKN+09] Alexander Lenk, Markus Klems, Jens Nimis, Stefan Tai, and Thomas Sandholm. Whats inside the cloud? an architectural map of the cloud landscape. In *IEEE Cloud '09*, 2009.

[MCC04] Matthew L. Massie, Brent N. Chun, and David E. Culler. The ganglia distributed monitoring system: design, implementation, and experience. In *Journal of Parallel Computing*, April 2004.

[MGT10] M. Morgan, A.S. Grimshaw, and O. Tatebe. Rns specification. Web, 2010. Available from: `http://www.ogf.org/Public\_Comment\_Docs/Documents/2010-03/RNS-spec-v11.pdf`.

[MKF11] P. Marshall, K. Keahey, and T. Freeman. Improving utilization of infrastructure clouds. In *CCGrid 2011*, 2011.

[Mor05] M Morgan. Byteio specification. Web, October 2005. Available from: `http://www.ogf.org/Public\_Comment\_Docs/Documents/Feb-2006/draft-byteio-rec-doc-v1-1.pdf`.

[MTF09] Andrea Matsunaga, Mauricio Tsugawa, and Jose Fortes. Cloudblast: Combining mapreduce and virtualization on distributed resources for bioinformatics applications. In *4th IEEE Intl Conference on eScience*, 2009.

[nsf09] Futuregrid: An experimental, high-performance grid test-bed. Web Page, 2009. October 1, 2009 - September 30, 2013 (Estimated). Available from: `http://www.nsf.gov/awardsearch/showAward.do?AwardNumber=0910812`.

[OAS07] OASIS. Ws-trust. Web, March 2007. Available from: `http://docs.oasis-open.org/ws-sx/ws-trust/v1.3/ws-trust.pdf`.

[OR02] Klaus-Dieter Oertel and Mathilde Romberg. The UNICORE Grid System, Tutorial 3, August 2002.

[pap12] PAPI, 2012. Available from: `http://icl.cs.utk.edu/papi/`.

[RTM+10] P. Riteau, M. Tsugawa, A. Matsunaga, J. Fortes, T. Freeman, D. LaBissoniere, , and K. Keahey. Sky computing on futuregrid and grid'5000. In *TeraGrid 2010*, 2010.

[sag] SAGA BigJob. Available from: `https://github.com/saga-project/BigJob/wiki`.

[sag12] Saga Project, 2012. Available from: `http://www.saga-project.org/`.

[SBC+03] Constantine P. Sapuntzakis, David Brumley, Ramesh Chandra, Nickolai Zeldovich, Jim Chow, Monica S. Lam, and Mendel Rosenblum. Virtual appliances for deploying and maintaining software. In *LISA*, pages 181–194. USENIX, 2003. Available from: `http://dblp.uni-trier.de/db/conf/lisa/lisa2003.html#SapuntzakisBCZCLR03`.

[sca12] ScaleMP, 2012. Available from: `http://www.scalemp.com/`.

[SEHO07] Shava Smallen, Kate Ericson, Jim Hayes, and Catherine Olschanowsky. User-level grid monitoring with inca 2. In *Proceedings of the 2007 workshop on Grid monitoring*, GMW '07, pages 29–38, New York, NY, USA, 2007. ACM. Available from: `http://doi.acm.org/10.1145/1272680.1272687`.

[SNA07] SNAPP - SNMP Network Analysis and Presentation Package. `http://snapp.sourceforge.net/`, 2007.

[Sne03] David Snelling. *Unicore and the Open Grid Services Architecture*, pages 701–712. John Wiley & Sons, Ltd, 2003. Available from: `http://dx.doi.org/10.1002/0470867167.ch29`.

[SOHL+98] Marc Snir, Steve Otto, Steven Huss-Lederman, David Walker, and Jack Dongarra. *MPI-The Complete Reference, Volume 1: The MPI Core*. MIT Press, Cambridge, MA, USA, 2nd. (revised) edition, 1998.

[tak12] TakTuk, 2012. Available from: `http://taktuk.gforge.inria.fr/`.

[TF06] Mauricio Tsugawa and Jose A. B. Fortes. A virtual network (vine) architecture for grid computing. In *20th Intl. Parallel and Distributed Processing Symposium (IPDPS)*, 2006.

[TJC+98] Brian Tierney, William Johnston, Brian Cowley, Gary Hoo, Chris Brooks, and Dan Gunter. The netlogger methodology for high performance distributed systems performance analysis. In *In Proc. 7th IEEE Symp. on High Performance Distributed Computing*, pages 260–267, 1998.

[vam12] Vampir, 2012. Available from: `http://www.vampir.eu/`.

[VDRB11] J.-S. Vöckler, E. Deelman, M. Rynge, and G.B. Berriman. Experiences using cloud computing for a scientific workflow application. In *Workshop on Scientific Cloud Computing (ScienceCloud)*, June 2011.

[vL06] Gregor von Laszewski. Java CoG Kit Workflow Concepts. *Journal of Grid Computing*, 3(3-4):239–258, January

2006. Available from: http://www.mcs.anl.gov/~gregor/papers/vonLaszewski-workflow-taylor-anl.pdf.

[vLDWF12] Gregor von Laszewski, Javier Diaz, Fugang Wang, and Geoffrey C. Fox. Comparison of multiple cloud frameworks. In *IEEE Cloud 2012*, Honolulu, 2012.

[vLFW+10] Gregor von Laszewski, Geoffrey C. Fox, Fugang Wang, Andrew J Younge, Archit Kulshrestha, Gregory G. Pike, Warren Smith, Jens Voeckler, Renato J. Figueiredo, Jose Fortes, Kate Keahey, and Ewa Delman. Design of the futuregrid experiment management framework. In *Proceedings of Gateway Computing Environments 2010 (GCE2010) at SC10*, New Orleans, LA, 2010. IEEE. Available from: http://grids.ucs.indiana.edu/ptliupages/publications/vonLaszewski-10-FG-exp-GCE10.pdf.

[vLYH+09] Gregor von Laszewski, Andrew Younge, Xi He, Kumar Mahinthakumar, and Lizhe Wang. Experiment and Workflow Management Using Cyberaide Shell. In *4th International Workshop on Workflow Systems in e-Science (WSES 09) in conjunction with 9th IEEE International Symposium on Cluster Computing and the Grid*, pages 568–573, Shanghai, China, May 2009. IEEE. Available from: http://cyberaide.googlecode.com/svn/trunk/papers/09-gridshell-ccgrid/vonLaszewski-ccgrid09-final.pdf.

[WF11] David Wolinsky and Renato Figueiredo. Experiences with self-organizing, decentralized grids using the grid appliance. In *Proceedings of the 20th International ACM Symposium on High-Performance Parallel and Distributed Computing (HPDC-2011)*, San Jose, CA, June 2011. ACM.

[wwwa] Apache Hadoop! Webpage. Available from: http://hadoop.apache.org/.

[wwwb] FutureGrid Portal. Webpage. Available from: http://portal.futuregrid.org.

[wwwc] Nimbus Project. Available from: http://www.nimbusproject.org/.

[wwwd] Open Cloud Computing Interface (OCCI). Webpage. Available from: http://occi-wg.org/.

[wwwe] Open Grid Forum. Webpage. Available from: http://www.ogf.org/.

[wwwf] Open Source Eucalyptus. Webpage. Available from: http://open.eucalyptus.com/.

[wwwg] OpenNebula. Webpage. Available from: `http://opennebula.org/`.

[wwwh] OpenStack. Webpage. Available from: `http://openstack.org/`.

[wwwi] Simple Linux Utility for Resource Management (SLURM). Webpage. Available from: `https://computing.llnl.gov/linux/slurm/`.

[wwwj] xCAT Extreme Cloud Administration Toolkit. Webpage. Available from: `http://xcat.sourceforge.net/`.

[xse12] XSEDE - Extreme Science and Engeneering Environment, 2012. Available from: `http://www.xsede.org`.

Index

Accounting, 27
Authentication, 26
Authorization, 26

Cloud, 2
    Accounting, 27
    Sensor, 33

Dynamic Provisioning, 20
Dynamic provisioning, 15

Education, 37
Eucalyptus, 12
Experiment management, 24
    DevOps-based, 26
    Interactive, 25
    RAIN-Based, 26
    Workflow-based, 25

FutureGrid, 1
    Architecture, 8
    Experiment management, 24
    Image management, 15
    Network Impairment, 4
    RAIN, 15
    Resources
        alamo, 5
        bravo, 5
        delta, 5
        foxtrot, 5
        hotel, 5
        india, 5
        sierra, 5
        xray, 5

Ganglia, 24
Grid services, 14

HPC accounting, 27
HPC services, 14

IaaS, 11
Image generation, 17
Image management, 15
Image registration, 19
Image repository, 18
Inca, 23
Information service, 22
Infrastructure as a Service, 11
Intelligent resource adaptations, 27

LDAP, 26

MapReduce, 13, 32
Meta-scheduler, 27
Monitoring service, 22

Netlogger, 23
Nimbus, 11
    infrastructure, 11
    platform, 12

OpenStack, 12
Outreach, 37

PaaS, 13
PAPI, 15
Performance service, 22
perfSONAR, 24
Platform as a Service, 13
Portal, 28
Project management, 24

RAIN, 15, 20, 26

SAGA, 32

SNAPP, 24
Symmetric Multiprocessing, 14

User Level Virtual Networks, 12

Vampir, 15
ViNe, 12