USING CLOUDS FOR SCIENCE, IS IT JUST KICKING THE CAN DOWN THE ROAD?

Ewa Deelman[1], Gideon Juve[1] and G. Bruce Berriman[2]
[1]*Information Sciences Institute, University of Southern California, Marina del Rey, CA, U.S.A.*
[2]*Infrared Processing and Analysis Center, California Institute of Technology, Pasadena, CA, U.S.A.*

Keywords: Scientific Workflows, Grids and Clouds, Cloud Configuration, Application Management.

Abstract: In this paper we describe issues related to the execution of scientific workflows on clouds, giving particular emphasis to the challenges faced by scientists when using grids and clouds for workflows. We also mention some existing solutions and identify areas requiring additional work.

1 INTRODUCTION

Over the past decade grid computing has been put forward as a platform for conducting science. We have seen both tightly and loosely coupled parallel applications making use of national and international infrastructures such as the Open Science Grid (OSG) (www.opensciencegrid.org), TeraGrid (http://www.teragrid.org/), EGI (Kranzlmüller et al., 2010), and others. These applications have been developed for domains such as astronomy (Deelman et al., 2008), bioinformatics (Stevens et al., 2003), earthquake science (Callaghan et al., 2008), physics (Brown et al., 2006), and others.

The broad spectrum of distributed computing provides unique opportunities for large-scale, complex scientific applications in terms of resource selection, performance optimization, reliability, etc. However, many issues of usability, reliability, performance, and efficient computation remain challenges in the area of grid computing. Over time the architecture of the underlying clusters used for scientific computation has been changing, from commodity-type architectures connected by high-performance networks to more complex architectures and deployments such as those of the Cray XT5 (Bland et al., 2009) or IBM Blue Gene (Gara et al., 2005). As the computing architectures have changed, so has the software used to access these machines across the wide area networks. For example, the Globus Toolkit (Foster, 2006) has undergone many revisions and architectural changes over the past decade.

Although the various hardware and software systems have been changing, users have been dealing with the same issues when executing applications on these distributed systems. One of these issues is usability. Much of the software that is deployed today is very complex and often geared towards high-end users that have already invested a large amount of time and effort to learn the ins-and-outs of the cyberinfrastructure. An average scientist who is ready to scale up his or her computations to the large-scale infrastructure is faced with learning new software components that will move the data to the computations, schedule the jobs, and bring the results back. Although these functions seem relatively straight forward, they involve relying on a stack of software that includes services (e.g. Condor (Litzkow et al., 1988) /GRAM (Czajkowski et al., 1998) /PBS (Bayucan et al., 1999) /local OS) for scheduling jobs and a data transfer service (e.g. GridFTP (Allcock et al., 2001)) that is interacting with a parallel file system (e.g. Lustre (Sun Microsystems), PVFS, etc.) deployed on the resource. The software stack can be unreliable and prone to user, system, and software errors. If an error occurs in that software stack, then it is very hard for the user to discover the source of the problem. Currently there are no debugging tools available to enable the tracing of the problem. Tuning the system for performance is an additional concern. Each of the system components has their own controls, and exposes only some of them to the users. As a result, obtaining performance from an application can be difficult.

In addition to the large-scale cyberinfrastructure, applications can target campus clusters, or utility computing platforms such as commercial (Amazon Elastic Compute Cloud); (Google App Engine) and academic clouds (Nimbus Science Cloud); (FutureGrid). However, these opportunities also bring with them their own set of challenges. It is hard to decide which resources to use and how long they will be needed. It is hard to determine what the cost/benefit trade-offs are when running in a particular environment. It is also difficult to achieve good performance and reliability for an application on a given system.

In the paper, we describe an approach to managing scientific applications, in particular scientific workflows, on cloud resources. We use this example as an illustration of the challenges faced by such applications on clouds, and postulate that although cloud computing is, in some ways, an improvement in distributed computing capabilities, many issues still need to be addressed.

2 APPLICATION CHALLENGES

Scientific workflows are being used today in a number of disciplines. They stitch together computational tasks so that they can be executed automatically and reliably on behalf of the researcher. For example, in astronomy, researchers use workflows to generate science-grade mosaics of the sky (Berriman et al., 2004); (http://montage.ipac.caltech.edu). These workflows are composed of a number of image processing applications that discover the geometry of the input images on the sky, calculate the geometry of the output mosaic on the sky, re-project the flux in the input images to conform to the geometry of the output mosaic, model the background radiation in the input images to achieve common flux scales and background levels across the mosaic, and rectify the background that makes all constituent images conform to a common background level. Finally these normalized images are added together to form the final mosaic.

The search for exoplanets is another example. The NASA Kepler mission (http://kepler.nasa.gov/) uses high-precision photometry to search for transiting exoplanets around main sequence stars. In May 2009, it began a photometric transit survey of 170,000 stars that has a nominal mission lifetime of 3.5 years. By the end of 2010, the Kepler mission had released light curves of 210,664 stars; these light curves contain measurements made over 229 days, with between 500 to 50,000 epochs per light curve. Analysing these light curves to identify periodic signals, such as those that arise from transiting planets and from stellar variability, requires calculations of periodograms that reveal periodicities in time-series data and estimates of their significance. Because periodograms require large amounts of computation to produce, workflows are being used to coordinate periodogram generation across distributed computing infrastructures.

Another example is from the earthquake science domain, where researchers use workflows to generate earthquake hazard maps of the Southern California region (W. G. et al., 2006). These maps show the maximum seismic shaking that can be expected to happen in a given region over a period of time (typically 50 years). Each point is obtained from a single hazard curve and each curve is generated by a workflow containing ~800,000 to ~1,000,000 computational tasks (Callaghan et al., 2008). This application requires large-scale computing capabilities such as those provided by the NSF TeraGrid (http://www.teragrid.org/).

In order to support such workflows, software systems need to 1) adapt the workflows to the execution environment (which, by necessity, is often heterogeneous and distributed), 2) optimize workflows for performance to provide a reasonable time to solution, 3) provide reliability so that scientists do not have to manage the potentially large numbers of failures that may occur, and 4) manage data so that it can be easily found and accessed at the end of the execution.

2.1 Cloud Issues

Having a number of capabilities, clouds can provide a variety of different solutions for applications. The latter have a choice of either adapt themselves to the new computational models provided by the cloud (such as MapReduce (Dean and Ghemawat, 2008)) or adapt the cloud to look like a computational environment that the applications have used so far—a compute cluster. Although new applications, especially those in bioinformatics, are willing to adopt new programming models, other applications, which have been using long time-validated and community accepted codes, are not willing to re-write their applications.

Virtualization in general opens up a greater number of resources to legacy applications. These applications are often very brittle and require a very specific software environment to execute successfully. Today, scientists struggle to make the codes that they rely on for weather prediction, ocean modelling, and many other computations work on different execution sites. No one wants to touch the codes that have been designed and validated many years ago in fear of breaking their scientific quality.

Clouds and their use of virtualization technologies may make these legacy codes much easier to run. With virtualization the environment can be customized with a given OS, libraries, software packages, etc. The needed directory structure can be created to anchor the application in its preferred location without interfering with other users of the system. The downside is obviously that the environment needs to be created and this may require more knowledge and effort on the part of the scientist than they are willing or able to spend.

For cluster-friendly applications, clouds can be configured (with additional work and tools) to look like a remote cluster, presenting interfaces for remote job submission and data transfer. As such, scientists can use existing grid software and tools to get their work done.

Another interesting aspect of the cloud is that, by default, it includes resource provisioning as part of the usage mode. Unlike the grid, where jobs are often executed on a best-effort basis, when running on the cloud, a user requests a certain amount of resources and has them dedicated for a given duration of time. Resource provisioning is particularly useful for workflow-based applications, where overheads of scheduling individual, inter-dependent tasks in isolation (as it is done by grid clusters) can be very costly. For example, if there are two dependent jobs in the workflow, the second job will not be released to a local resource manager on the cluster until the first job successfully completes. Thus the second job will incur additional queuing delays. In the provisioned case, as soon as the first job finishes, the second job is released to the local resource manager and since the resource is dedicated, it can be scheduled right away. Thus the overall workflow can be executed much more efficiently.

3 MANAGING WORK ON CLOUDS

3.1 Application Management

One approach to managing workflow-based applications in grid or cloud environments is to use the Pegasus Workflow Management System (Deelman et al., 2004). Pegasus runs scientific workflows on desktops, private clusters, campus clusters, the Open Science Grid (www.openscience grid.org), the TeraGrid (http://www.teragrid.org/), and academic and commercial clouds (Amazon Elastic Compute Cloud); (Google App Engine); (Nimbus Science Cloud). Pegasus can be used to run workflows ranging from just a few computational tasks to a million tasks. When errors occur, Pegasus tries to recover when possible by retrying tasks, by retrying the entire workflow, by providing workflow-level check-pointing, by re-mapping portions of the workflow, by trying alternative data sources for staging data, and, when all else fails, by providing a rescue workflow containing a description of only the work that remains to be done so that the user can resubmit it later when the problem is resolved (Deelman et al., 2006). Pegasus cleans up storage as the workflow is executed so that data-intensive workflows have enough space to execute on storage-constrained resources (Ramakrishnan et al., 2007); (Singh et al., 2007). Pegasus keeps track of what has been done (provenance) including the locations of data used and produced, and which software was used with which parameters (Miles et al., 2007); (Miles et al., 2008); (Groth et al., 2009).

Pegasus assumes that the workflow management system and the workflow description live on a *submit host*, which is under the user's control. Pegasus launches jobs from the submit host to the execution sites (either local or remote). The notion of a submit host and execution environment lends itself well to the Infrastructure as a Service (IaaS) model of cloud computing (Deelman, 2009) .

Although Pegasus controls workflow execution through both static workflow restructuring and dynamic tuning of the job execution, it does not control the resources where the jobs are executing. However, managing resources is important in grids when trying to optimize workflow performance and is critical in clouds where the resources are not necessarily preconfigured for the workflow. As a result, we advocate the approach of building virtual clusters on the cloud and configuring them in a way similar to the clusters encountered by grid applications.

3.2 Building Virtual Clusters

Deploying applications in the cloud is not a trivial task. It is usually not sufficient to simply develop a virtual machine (VM) image that runs the appropriate services when the virtual machine starts up, and then just deploy the image on several VMs in the cloud. Often, the configuration of distributed services requires information about the nodes in the deployment that is not available until after nodes are provisioned (such as IP addresses, host names, etc.) as well as parameters specified by the user. In addition, nodes often form a complex hierarchy of interdependent services that must be configured in the correct order. Although users can manually configure such complex deployments, doing so is time consuming and error prone, especially for deployments with a large number of nodes. Instead, we advocate an approach where the user is able to specify the layout of their application declaratively, and use a service to automatically provision, configure, and monitor the application deployment.

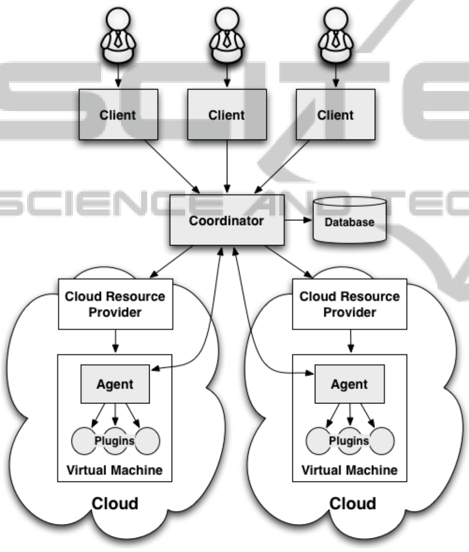

Figure 1: Wrangler architecture.

The service should allow for the dynamic configuration of the deployment, so that a variety of services can be deployed based on the needs of the user. It should also be resilient to failures that occur during the provisioning process and allow for the dynamic addition and removal of nodes.

The Wrangler system (Juve and Deelman, 2011); (Juve and Deelman, 2011) allows users to send a simple XML description of the desired deployment to a web service that manages the provisioning of virtual machines and the installation and configuration of software and services. It is capable of interfacing with many different resource providers in order to deploy applications across clouds, supports plugins that enable users to define custom behaviors for their application, and allows dependencies to be specified between nodes. Complex deployments can be created by composing several plugins that set up services, install and configure application software, download data, and monitor services on several interdependent nodes.

The components of the system are shown in Figure 1. They include: *clients*, a *coordinator*, and *agents*.

- *Clients* run on each user's machine and send requests to the coordinator to launch, query, and terminate, deployments. Clients have the option of using a command-line tool, a Python API, or XML-RPC to interact with the coordinator.
- The *coordinator* is a web service that manages application deployments. It accepts requests from clients, provisions nodes from cloud providers, collects information about the state of a deployment, and acts as an information broker to aid application configuration. The coordinator stores information about its deployments in an SQLite database.
- *Agents* run on each of the provisioned nodes to manage their configuration and monitor their health. The agent is responsible for collecting information about the node (such as its IP addresses and hostnames), reporting the state of the node to the coordinator, configuring the node with the software and services specified by the user, and monitoring the node for failures.
- *Plugins* are user-defined scripts that implement the behavior of a node. They are invoked by the agent to configure and monitor a node. Each node in a deployment can be configured with multiple plugins.

Users specify their deployment using a simple XML format. Each XML request document describes a deployment consisting of several *nodes*, which correspond to virtual machines. Each node has a *provider* that specifies the cloud resource provider to use for the node, and defines the characteristics of the virtual machine to be provisioned — including the VM image to use and the hardware resource type — as well as authentication credentials required by the provider. Each node has one or more *plugins*, which define the behaviors, services, and functionality that should be implemented by the node. Plugins can have multiple *parameters*, which enable the user to configure the plugin, and are passed to the script when it is executed on the node. Nodes may be members of a named *group*, and each node may depend on zero or more other nodes or groups.

3.3 Observation of Cloud Failures

We can use Wrangler to build virtual clusters across heterogeneous clouds. In the example below we ran the test workflow 3 times provisioning 6 nodes (or 48 cores on each resource): once on FutureGrid (Sierra host), an academic cloud deployment in the US, once on Magellan, which was an experimental cloud deployment supported by the US Department of Energy, and once on both Sierra and Magellan at the same time using 6 nodes (48 cores) each (for a total of 96 cores). Figure 2 illustrates the deployment, which consists of a Master Node (submit host) and a virtual cluster running Condor provisioned by Wrangler.

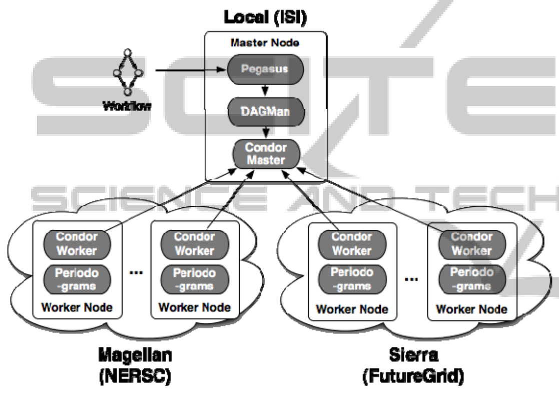

Figure 2: Virtual cluster using local resources, and cloud resources from both Sierra and Magellan, that was used to execute the periodograms workflows.

After some failures we were able to execute 3 complete runs of the periodogram workflow, which read a total of 5.4 GB of input data, produced 30 GB of output data, and consumed 713 hours of CPU time, all within the course of a few hours.

Although Wrangler was able to successfully provision resources and execute the application, we found that failures were a major problem. We encountered several different types of failures while running this application. Table 1 shows a breakdown of the number of requests made and the number and type of failures for both Sierra and Magellan. The failures we encountered include:

- **Failed to start**: The request was accepted, but the VM state went from 'pending' to 'terminated' without reaching 'running'.
- **No route to host**: The VM is in the 'running' state and reports to the Coordinator, but does not respond to network connections, or it stops responding to network connections after some time.
- **Invalid IP**: The provider's information service reported a public IP of '0.0.0.0' for the VM.
- **No public IP**: VM was assigned two private IP addresses instead of one public and one private.
- **Request timed out**: Resource provider's web service hanged and the connection timed out. These failures were automatically corrected by retrying the request.
- **Insufficient resources**: The resource provider was not able to satisfy the request due to the lack of available resources.

The failure rate we observed in running this application was significant. Five out of 20 requests to Sierra failed, which translates to a failure rate of 25%. Magellan was even worse with 12 out of 28, or 42%, of requests failing. The high failure rate on Magellan was primarily due to a network outage that caused several VMs to fail at the same time, and several failures were caused by a lack of resources, which is not a failure in the strictest sense. The effect of each of these failures is a request that did not result in a usable VM. Although these failures may, in some cases, be prevented by changes to the cloud management system, the fact is that such failures will occasionally occur. Provisioning systems like Wrangler should be prepared to detect and manage such failures, otherwise it will be difficult for an application to achieve meaningful results.

We found that using Wrangler greatly simplified the process of provisioning and configuring cloud resources to execute this application. We were able to provision resources across two different clouds, quickly deploy a resource management system, and perform a non-trivial amount of computation in a short amount of time. However, we determined that more work is needed to automatically detect and recover from failures.

Table 1: Number of requests and number and type of failures for the periodograms application on Sierra and Magellan.

	FutureGrid/Sierra	**NERSC/ Magellan**
Requests	20	28
Failed to start	2	0
No route to host	0	5
Invalid IP	1	0
No public IP	1	0
Request timed out	1	1
Insufficient resources	0	6
Total Failures	**5**	**12**

4 RELATED WORK

There has been a considerable amount of work in the area of scientific workflows (Deelman et al., 2009). Here we just present related work in the area of virtual clusters.

Constructing clusters on top of virtual machines has been explored by several previous research efforts. These include VMPlants (Krsul et al., 2004), StarCluster (http://web.mit.edu/stardev/cluster/), and others (Murphy et al., 2009); (Vöckler et al., 2011). These systems typically assume a fixed architecture that consists of a head node and N worker nodes. They also typically support only a single type of cluster software, such as SGE, Condor, or Globus.

Many different configuration management and policy engines have been developed for UNIX systems. Cfengine (Burgess, 1995), Puppet (Kanies, 2006), and Chef (http://www.opscode.com/chef) are a few well-known examples. Wrangler is similar to these systems in that configuration is one of its primary concerns, however, the other concern of this work, provisioning, is not addressed by configuration management systems.

This work is related to virtual appliances (Sapuntzakis et al., 2003) in that we are interested in deploying application services in the cloud. The focus of our project is on deploying collections of appliances for distributed applications. As such, our research is complementary to that of the virtual appliances community.

Wrangler is similar to the Nimbus Context Broker (NCB) (http://workspace.globus.org) used with the Nimbus cloud computing system (Keahey et al., 2008). NCB supports *roles*, which are similar to Wrangler plugins with the exception that NCB roles must be installed in the VM image and cannot be defined by the user when the application is deployed. In addition, our system is designed to support multiple cloud providers, while NCB works best with Nimbus-based clouds.

Recently, other groups are recognizing the need for deployment services, and are developing similar solutions. One example is cloudinit.d (Bresnahan et al., 2011), which enables users to deploy and monitor interdependent *services* in the cloud. Cloudinit.d services are similar to Wrangler plugins, but each node in cloudinit.d can have only one service, while Wrangler enables users to compose several, modular plugins to define the behavior of a node.

5 DISCUSSION AND CONCLUSIONS

We have shown that workflow-based applications can run successfully on cloud resources using the same execution model as they use on the grid. However, there are still many obstacles to making this mode of execution efficient and robust. Although cluster configuration tools exist, they need to be able to deal with the failures we see in the underlying cloud software. A big challenge, just as is in the case of grids, is managing the failures, either by masking them, by presenting them to the user in an understandable way, and/or by providing tools to help pinpoint the source of problems.

If one approaches application execution the way we do, where we stand up a cloud infrastructure that is similar to what can be found on campus or national resources, then we still have the same problems that we see in grids, with a number of software systems layered on top of each other. We still have issues of deciphering problems when errors are not passed gracefully between the software layers. Just as with grids, there are no debugging tools or sophisticated user-friendly monitoring tools for applications running on cloud environments. Although virtualization can be a very powerful tool for providing better reliability and performance, today’s tools do not take full advantage of it.

Although commercial clouds, such as Amazon, are currently much more reliable than their academic counterparts, there are monetary costs associated with their use. Therefore applications developers and users need tools to help evaluate the cost and turnaround time of the entire computational problem (for example whole sets of workflows—ensembles). They also need tools to manage these costs for example by using cost-effective resources or leveraging allocations on grid systems.

For a number of new bioinformatics applications, which are entering the arena of cloud computing, issues of data privacy and security are critical. Thus a new understanding and evaluation of the security practices in virtual environments needs to be developed.

ACKNOWLEDGEMENTS

This work was supported by the National Science Foundation under grant OCI-0943725.

G. B. Berriman is supported by the NASA Exoplanet Science Institute at the Infrared Processing and Analysis Center, operated by the California Institute of Technology in coordination with the Jet Propulsion Laboratory (JPL).

REFERENCES

"Open Science Grid," www.opensciencegrid.org.

TeraGrid. Available: http://www.teragrid.org/

D. Kranzlmüller, J. M. Lucas, et al, "The European Grid Initiative (EGI)," *Remote Instrumentation and Virtual Laboratories,* pp. 61-66, 2010.

E. Deelman, G. Singh, et al, "The Cost of Doing Science on the Cloud: The Montage Example," SC'08 Austin, TX, 2008.

R. D. Stevens, A. J. Robinson, and C. A. Goble, "myGrid: personalised bioinformatics on the information grid," *Bioinformatics* vol. 19, 2003.

S. Callaghan, P. Maechling, et al, "Reducing Time-to-Solution Using Distributed High-Throughput Mega-Workflows - Experiences from SCEC CyberShake," e-Science, Indianapolis, 2008.

D. A. Brown, P. R. Brady, et al, "A Case Study on the Use of Workflow Technologies for Scientific Analysis: Gravitational Wave Data Analysis," in *Workflows for e-Science,* I. Taylor, et al, Eds., Springer, 2006.

A. S. Bland, R. A. Kendall, et al, "Jaguar: The world's most powerful computer," *Memory (TB),* vol. 300, p. 362, 2009.

A. Gara, M. A. Blumrich, et al, "Overview of the Blue Gene/L system architecture," *IBM Journal of Research and Development,* vol. 49, 2005.

I. Foster, "Globus Toolkit Version 4: Software for Service-Oriented Systems," 2006.

M. Litzkow, M. Livny, and M. Mutka, "Condor - A Hunter of Idle Workstations," in *Proc. 8th Intl Conf. on Distributed Computing Systems,* ed, 1988.

K. Czajkowski, I. Foster, et al, "A Resource Management Architecture for Metacomputing Systems," in *4th Workshop on Job Scheduling Strategies for Parallel Processing,* 1998, pp. 62-82.

A. Bayucan, R. L. Henderson, et al, "Portable Batch System: External reference specification," ed, 1999.

W. Allcock, J. Bester, et al, "Data Management and Transfer in High-Performance Computational Grid Environments," *Parallel Computing,* 2001.

Sun Microsystems. *Lustre*. http://www.lustre.org

Amazon Elastic Compute Cloud. http://aws.amazon.com/ec2/

Google App Engine. http://code.google.com/appengine/

Nimbus Science Cloud. http://workspace.globus.org/clouds/nimbus.html

(2010). *FutureGrid*. http://www.futuregrid.org/

G. B. Berriman, E. Deelman, et al, "Montage: A Grid Enabled Engine for Delivering Custom Science-Grade Mosaics On Demand," in *SPIE Conference 5487: Astronomical Telescopes,* 2004.

Montage. Available: http://montage.ipac.caltech.edu

R. W. G., Paul G. Somerville, et al, "Ground motion environment of the Los Angeles region," *The Structural Design of Tall and Special Buildings,* vol. 15, pp. 483-494, 2006.

J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Communications of the ACM*, vol. 51, pp. 107-113, 2008.

E. Deelman, J. Blythe, et al, "Pegasus : Mapping Scientific Workflows onto the Grid," in *2nd European Across Grids Conference*, Cyprus, 2004.

E. Deelman, G. Mehta, et al, "Pegasus: Mapping Large-Scale Workflows to Distributed Resources," in *Workflows in e-Science*, I. Taylor, E. Deelman, D. Gannon, and M. Shields, Eds., ed: Springer, 2006.

A. Ramakrishnan, G. Singh, et al, "Scheduling Data -Intensive Workflows onto Storage-Constrained Distributed Resources," in *CCGrid* 2007.

G. Singh, K. Vahi, et al, "Optimizing Workflow Data Footprint " *Scientific Programming Journal, Special issue on Dynamic Computational Workflows,* vol. 15, 2007

S. Miles, E. Deelman, et al, "Connecting Scientific Data to Scientific Experiments with Provenance " Third IEEE e-Science 2007, India. , 2007.

S. Miles, P. Groth, et al, "Provenance: The bridge between experiments and data," *Computing in Science & Engineering,* vol. 10, pp. 38-46, 2008.

P. Groth, E. Deelman, et al, "Pipeline-Centric Provenance Model, "The 4th Workshop on Workflows in Support of Large-Scale Science, Portland, OR, 2009.

E. Deelman, "Grids and Clouds: Making Workflow Applications Work in Heterogeneous Distributed Environments," *International Journal of High Performance Computing Applications,* 2009.

G. Juve and E. Deelman, "Automating Application Deployment in Infrastructure Clouds," CloudCom 2011,

G. Juve and E. Deelman, "Wrangler: Virtual Cluster Provisioning for the Cloud (short paper), HPDC'11, 2011.

E. Deelman, D. Gannon, et al, "Workflows and e-Science: An overview of workflow system features and capabilities," *Future Generation Computer Systems,* vol. 25, pp. 528-540, 2009.

I. Krsul, A. Ganguly, et al, "Vmplants: Providing and managing virtual machine execution environments for grid computing," 2004, pp. 7-7.

StarCluster. http://web.mit.edu/stardev/cluster/.

M. A. Murphy, B. Kagey, et al, "Dynamic provisioning of virtual organization clusters," 2009,

J.-S. Vöckler, G. Juve, et al, "Experiences Using Cloud Computing for A Scientific Workflow Application," ScienceCloud, 2011.

M. Burgess, "A site configuration engine," *USENIX Computing Systems,* vol. 8, 1995.

L. Kanies, "Puppet: Next Generation Configuration Management," *Login,* vol. 31, 2006.

Opscode, Chef. Available: http://www.opscode.com/chef.

C. Sapuntzakis, D. Brumley, et al, "Virtual Appliances for Deploying and Maintaining Software," USENIX 2003.

Nimbus. Available: http://workspace.globus.org

K. Keahey, R. Figueiredo, et al, "Science clouds: Early experiences in cloud computing for scientific applications," *Cloud Computing and Applications*, 2008.

J. Bresnahan, T. Freeman, et al, "Managing Appliance Launches in Infrastructure Clouds," Teragrid Conference, 2011.