This article has been accepted for publication in Computing in Science and Engineering but has not yet been fully edited.
Some content may change prior to final publication.

Comparing FutureGrid, Amazon EC2, and Open Science Grid for Scientific Workflows

Gideon Juve[1], Mats Rynge[1], Ewa Deelman*[1], Jens-S. Vöckler[1] and G. Bruce Berriman[2]

[1] Information Sciences Institute, University of Southern California, Marina Del Rey, CA 90292, USA
[2] Infrared Processing and Analysis Center, California Institute of Technology, Pasadena, CA 91125, USA
Email: {gideon, rynge, deelman*, voeckler}@isi.edu; gbb@ipac.caltech.edu;
* Corresponding author

Abstract

Scientists have a number of computing infrastructures available to conduct their research, including grids and public or private clouds. This paper explores the use of these cyberinfrastructures to execute scientific workflows, an important class of scientific applications. It examines the benefits and drawbacks of cloud and grid systems using the case study of an astronomy application. The application analyzes data from the NASA Kepler mission in order to compute periodograms, which help astronomers detect the periodic dips in the intensity of starlight caused by exoplanets as they transit their host star. In this paper we describe our experiences modeling the periodogram application as a scientific workflow using Pegasus, and deploying it on the FutureGrid scientific cloud testbed, the Amazon EC2 commercial cloud, and the Open Science Grid. We compare and contrast the infrastructures in terms of setup, usability, cost, resource availability and performance.

Keywords: Grid Computing, Cloud Computing, Scientific Workflows, Amazon EC2, Open Science Grid, FutureGrid, Astronomy

1. Introduction

As scientific data volumes grow, scientists will increasingly require data processing services that support easy access to distributed computing platforms such as grids and clouds. The astronomy community is undertaking surveys of unprecedented depth to understand the behavior of astrophysical sources with time, so much so that the 2010 Decadal Survey of Astronomy [4], which recommends national priorities for the field, declared the time domain "the last frontier" in astrophysics. These surveys are already beginning to produce data sets too large to be analyzed by local resources, and will culminate near the end of this decade with the Large Synoptic Survey Telescope (LSST) [33], which expects to produce petabyte-scale data sets each night.

Scientific workflows are a useful tool for managing these large-scale analyses because they express declaratively the relationships between computational tasks and data. As a result, workflows enable scientists to easily define multi-stage computational and data processing pipelines that can be executed in parallel on distributed resources, which automates complex analyses, improves application performance, and reduces the time required to obtain scientific results.

Because scientific workflows often require resources beyond those available on a scientist's desktop, it is important to understand what are the benefits and drawbacks of various cyberinfrastructures available to the user so that they can make informed decisions about the most suitable platform for their application. Today scientists have many different options for deploying their applications, including grids such as Open Science Grid [24] or XSEDE [8], commercial clouds such as Amazon EC2 [1] or The Rackspace Cloud [34] and academic clouds like FutureGrid [11]. Traditionally, scientific workflows have been executed on campus clusters, or grids. Recently, however, scientists have been investigating the use of commercial and academic clouds for these applications. In contrast to grids and other traditional HPC systems, which provide only best-effort service and are configured and maintained by resource

Digital Object Indentifier 10.1109/MCSE.2013.44 1070-9924/$26.00 2013 IEEE

owners, clouds provide a customizable infrastructure that enables scientists to provision resources and deploy the software environment they require on-demand.

Choosing the best platform for an application involves many tradeoffs in terms of effort, cost, and performance. In this paper we describe some of the opportunities and challenges of running scientific workflows on grids and commercial and academic clouds. We chose three execution infrastructures that are representative of each category: the Open Science Grid [24], Amazon EC2 [1], and FutureGrid [11]. We describe our experiences using these infrastructures for running a scientific workflow application that analyzes data from NASA's Kepler mission. We compare the infrastructures across a number of dimensions, including: obtaining an account, environment setup, cost, provisioning and resource availability.

2. Periodograms: An Astronomy Application

Our example workflow application processes time-series data collected by NASA's Kepler mission [19]. The Kepler satellite uses high-precision photometry to search for exoplanets transiting their host stars. In 2009 the Kepler mission began a multi-year transit survey of 170,000 stars near the constellation Cygnus. In 2010 the project released a dataset containing 210,664 light curves, which record how the brightness of a star changes over time.

Analyzing light curves to identify the periodic dips in brightness that arise from transiting exoplanets requires the calculation of *periodograms*, which reveal periodic signals in time-series data and estimate their significance. Generating periodograms is a computationally intensive process that requires high-performance, distributed computing.

In order to support the analysis of Kepler's 210,664 light curve dataset we developed a workflow using the Pegasus Workflow Management System [6]. The workflow, illustrated by Figure 1, consists of 43 sub-workflows, each of which contains approximately 5,000 tasks. Each task applies three different periodogram algorithms to an input light curve to produce three pairs of output files. Using estimates for the runtimes of the tasks, Pegasus uses runtime clustering to group the tasks in these workflows into 2,354 60-minute jobs that can be efficiently submitted to the target infrastructures. The workflow requires approximately 66 days of sequential computation, consumes 17 GB of input data, and produces 105 GB of output data in 1.2 million output files.

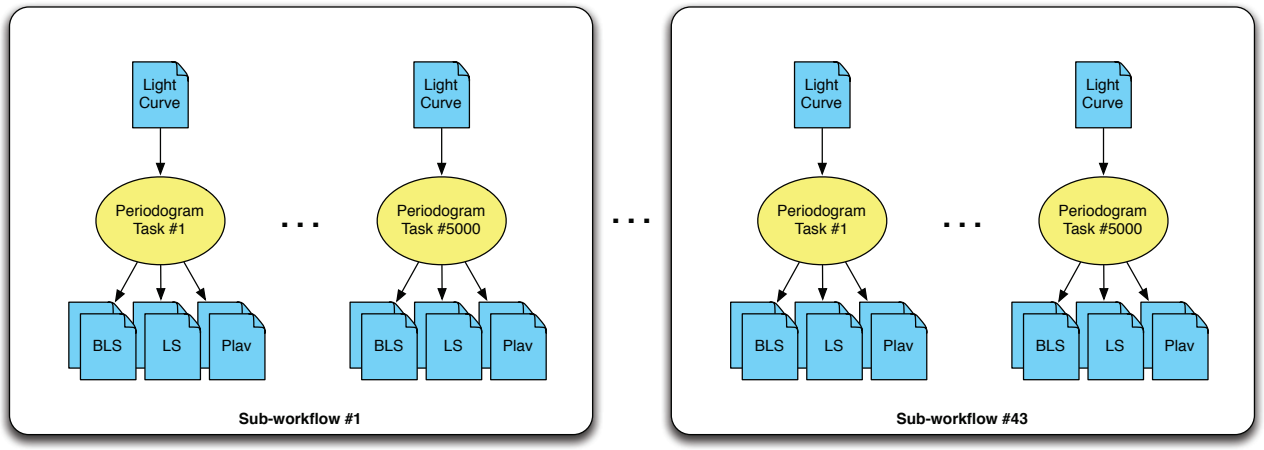

Figure 1: Kepler periodogram workflow of workflows with 43 sub-workflows containing ~5,000 tasks each.

Digital Object Indentifier 10.1109/MCSE.2013.44 1070-9924/$26.00 2013 IEEE

3. Execution Infrastructures

Figure 2 illustrates the workflow execution environment used with all three infrastructures. The workflow management system resides on a machine controlled by the workflow developer called the *submit host* (SH). The submit host maintains a queue of jobs that are ready to run on remote resources. The workflow is planned by Pegasus and submitted to DAGMan, which releases jobs in the correct order to Condor for execution on remote worker nodes. In our configuration, the submit host was located at ISI in Marina del Rey, CA for the cloud experiments, and at RENCI in North Carolina for the OSG experiments. For each job the input data was transferred automatically from the submit host to the remote workers, and the output data was transferred directly from the remote workers back to the submit host. Transfers of inputs and outputs were performed before and after each job. All files were compressed using gzip before being transferred, which reduced the amount of data transferred by more than 80%.

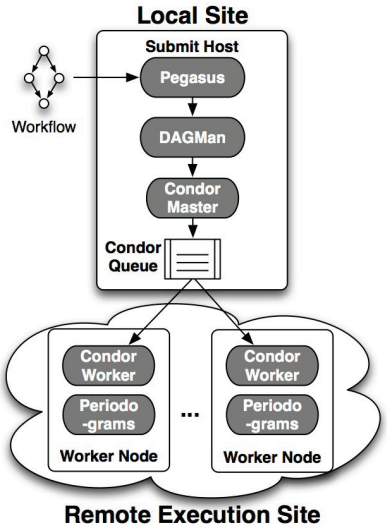

Figure 2: Overview of the execution environment.

3.1. Grid: Open Science Grid

The Open Science Grid (OSG) is a distributed computing consortium of major science experiments and academic institutions. There are currently approximately 80 sites offering compute and storage resources to OSG users. OSG is structured around virtual organizations (VOs) [9], which are logical groupings of people and experiments that have a common goal. VOs that own resources typically allow other VOs to use their resources on an opportunistic basis. This kind of sharing is one of the cornerstones in OSG. For example, the VO used for the experiments in this paper, the *Engagement VO*, helps new users get started and does not own any computational resources. Instead, Engagement users borrow unused resources from other VOs. The resource owners are free to set conditions on the acquisition of these resources and almost all give opportunistic jobs lower priority than their own jobs, and enable their jobs to preempt opportunistic jobs.

3.2. Commercial Cloud: Amazon EC2

The Amazon Elastic Compute Cloud (EC2) is a large commercial cloud operated by Amazon.com. EC2 is part of a suite of cloud services called Amazon Web Services (AWS). AWS services provide computational, storage, and communication infrastructure on-demand via web-based APIs. EC2 is a service for provisioning virtual machine instances from Amazon's datacenters. EC2 allows users to deploy VM images with customized operating systems, libraries, and application code on virtual machines with a variety of pre-defined hardware configurations (CPU,

Digital Object Indentifier 10.1109/MCSE.2013.44 1070-9924/$26.00 2013 IEEE

memory, disk) and prices. Users of EC2 are only billed for the resources they use and there are charges associated with VM instance hours and data transfers to and from Amazon's datacenter.

3.3. Academic Cloud: FutureGrid

The FutureGrid project [11] is designed to enable researchers to investigate issues related to the development and use of cloud computing systems, such as authentication, scheduling, virtualization, middleware, and cybersecurity.

FutureGrid includes a diverse set of distributed computing systems, storage, and a dedicated, high-bandwidth network. It supports a variety of open source cloud computing systems including Nimbus [20], Eucalyptus [21], and OpenStack [25], as well as "bare-metal" systems for experiments aimed at minimizing overhead and maximizing performance.

4. Infrastructure Setup

In this section we describe the means of getting access to the three infrastructures, and the steps needed to set up each environment for workflow execution.

4.1. Getting an Account

Creating an account for Amazon EC2 is simple, can be done entirely online, requires nothing more than a credit card, and takes no more than 15 minutes. Usage charges are billed to the user's credit card on a monthly basis. Once the account is created, users can download their security credentials and begin using Amazon's services immediately.

Gaining access to FutureGrid involves several steps. New users must first apply for an account, which takes one to three business days. Next, the user either joins an existing project, or creates a new project. Project proposal turnaround time is less than a week, and often just one business day. With an account and an approved project, a user can upload their public *SSH key* through the website and have access to resources within 24 hours. The entire process typically takes between a few days and a week.

On OSG, when a new user wants to join the Engagement VO, they first request an *X.509* user certificate from DOEGrids. Once the certificate request has been verified, and the certificate has been issued, the user can request membership in the Engagement VO using VOMS (VO Membership Service) and request an account on the community submit host. The entire process typically takes about a week.

4.2. Environment Setup

The procedure for creating the necessary execution environment on the cloud infrastructures (Amazon EC2 and FutureGrid) is very similar. The user needs to create an image that contains Condor, Pegasus, and any necessary application software and data. Installing these packages in the image reduces the setup time of the execution environment and reduces the amount of data that need to be transferred during workflow execution. It typically takes a few hours to create a single VM image, but learning the process can take a few days, or a few weeks if the user is not familiar with system administration. In our experience, maintaining VM images is one of the most tedious and troublesome parts of running a science application in the cloud. Scientists who are not familiar with system administration could find this aspect of infrastructure clouds to be a significant barrier.

OSG supports a range of options for accessing remote resources, from low-level grid APIs to web-based job management portals. The most common environment for new users consists of a community submit host that is pre-configured for executing workflows. For many scientists this approach is more attractive than a cloud deployment because the infrastructure software is installed and maintained by the VO, which can significantly reduce their startup time. For the experiments in this paper, we used a submit host provided by the Engagement VO. The host provides SSH access for users to log in and submit workloads using Pegasus. Compared to VMs in the cloud, where all the

Digital Object Indentifier 10.1109/MCSE.2013.44 1070-9924/$26.00 2013 IEEE

instances will look the same, OSG resources are much more heterogeneous and each resource may have a different software configuration. The Engagement VO provides a software overlay across the different resources to reduce the impact of this heterogeneity. Unlike the cloud case, neither the Pegasus tooling nor the application software are pre-staged to the resources, so the workflow has to stage those executables for each job.

4.3. Resource Provisioning

Given the complexity of the execution environments required to support distributed applications such as workflows, it is often necessary to use tools that support resource provisioning and configuration. These tools are used to allocate, configure, and manage resources by interacting with resource management systems, installing and configuring job management software, monitoring resources for failures, and deallocating resources that are no longer needed. Without these tools, deploying scientific workflows in both grids and clouds can be unsustainable.

Resources on Amazon EC2 were provisioned using Wrangler [13], a tool for deploying virtual clusters in the cloud. Wrangler users submit an XML description of their desired application deployment to a web service that manages the provisioning of resources and the installation and configuration of software and services. It 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 plugins to install and configure services on several interdependent nodes.

Resources on FutureGrid were provisioned manually from the Nimbus [20] cloud management system using the command-line client. Although Nimbus can automatically de-provision a VM after a user-specified interval, we chose to manually shutdown the VMs after finishing the experiment.

GlideinWMS [32] was used to provision resources on OSG. It has a distributed design with two main components: a *frontend* for each submit host and a *factory* for each compute resource. The frontend queries the Condor job queue to determine when the resource pool needs to grow. If an increase is desired, the frontend sends a request to the factory to provision more resources by submitting pilot jobs. For our OSG experiments the frontend was installed on the Engagement VO submit host at RENCI, and a community factory at UCSD was used for provisioning. Once the pilot jobs start up on a compute resource, they register back to the submit host at RENCI, which uses them to execute queued workflow jobs.

4.4. Cost

One of the most significant differences between Amazon EC2 and FutureGrid or OSG is cost. Amazon EC2 operates on a pay-as-you-go utility computing model where users are charged for the amount of computational resources they consume. In comparison, FutureGrid and OSG operate under an allocation model, common in academic computing, where users apply for access to computing resources and are allocated resources based on the scientific merit of their proposal. From the perspective of a scientist, academic resources such as FutureGrid and OSG are more attractive options for science applications than commercial clouds such as EC2 because they are effectively free.

Amazon has a very complex pricing model for EC2 and related services. In general, they charge per hour, per GB, and per operation across all their services. For the experiments in this paper we were charged for two services: data transfer and VM instances. Data transfer from our submit host to EC2 was free, but data transfer from EC2 back to our submit host was billed at $0.12/GB, or $2.44/workflow for the 20 GB of compressed output data (105 GB uncompressed) produced by the periodogram workflow. Amazon charges for EC2 VM instances by the hour and has two types of instance requests: on-demand instances, and spot instances. Users of spot instances bid the maximum amount they are willing to pay for resources, and as long as their bid is greater than or equal to the current spot price, the instances will be launched. Amazon periodically updates the current spot price based on the number of idle resources and the current bids using a proprietary algorithm. Typically, spot prices are significantly lower than on-demand prices. For the experiments in this paper we used spot instances for which we were charged an average of $0.24 per hour (a savings of

Digital Object Indentifier 10.1109/MCSE.2013.44 1070-9924/$26.00 2013 IEEE

65%). The total cost of spot instances for the periodogram application was approximately $61.44 per workflow (32 instances times 8 hours).

4.5. Resource Availability

Although the size of EC2 is not public information, it is known to have a very large number of resources. In 2011 Cycle Computing was able to provision a cluster with 30,000 cores by spreading requests across several datacenters [5]. In our experience we have always been able to provision all the resources we required for an application from EC2. For the experiments in this paper we requested 32 c1.xlarge spot instances (256 cores) several times with no failures or delays. In addition, our spot instances were never terminated early due to a change in spot prices.

Individual FutureGrid sites are relatively small, so we had to use resources from four different FutureGrid sites (Alamo, Sierra, Hotel, and Foxtrot) to get the required 256 cores. Although we were able to acquire the necessary resources for our experiment, FutureGrid resources are relatively scarce compared to Amazon EC2 and OSG.

The experience of obtaining resources on OSG is different depending on whether the user is part of a VO that owns resources or not. In the latter case, user jobs may have a low priority or be preempted if the resource owner submits more jobs. Most of the time, resource polices and system status are not available to the user, which means that opportunistic users will not know before submitting jobs if resources will be plentiful or scarce. In our experience the availability of OSG resources for opportunistic jobs varies significantly over time.

5. Experiments

This section describes the results of running the periodogram workflow on the three different infrastructures. The workflow was run several times on each infrastructure in order to detect and illustrate unusual behavior and failures. Figures 3-5 show the number of jobs over time for the most interesting runs of the experiment. The green part of each figure shows the number of concurrently running jobs. The grey curve, stacked on top of the green curve, shows the number of jobs in the Condor queue that are waiting for a resource to become idle.

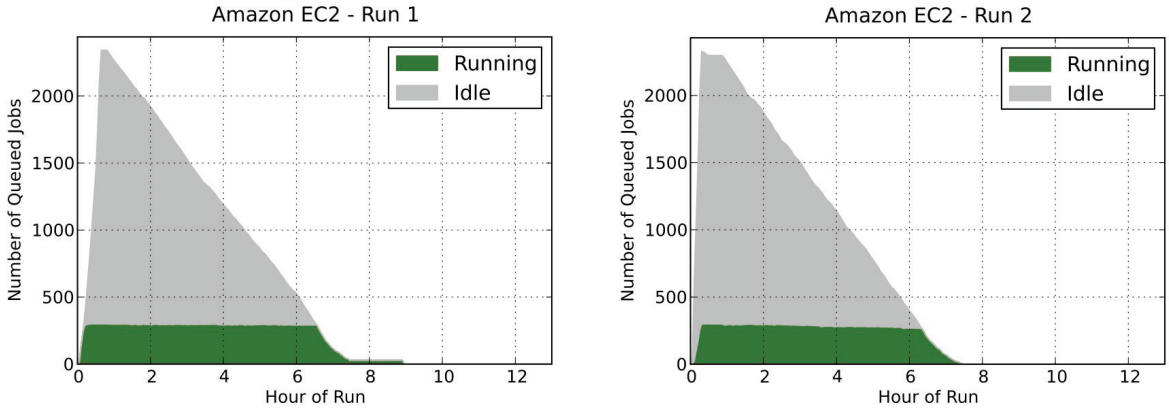

Figure 3: Periodogram runs on Amazon EC2 (Run 3 omitted).

Digital Object Indentifier 10.1109/MCSE.2013.44 1070-9924/$26.00 2013 IEEE

This article has been accepted for publication in Computing in Science and Engineering but has not yet been fully edited.
Some content may change prior to final publication.

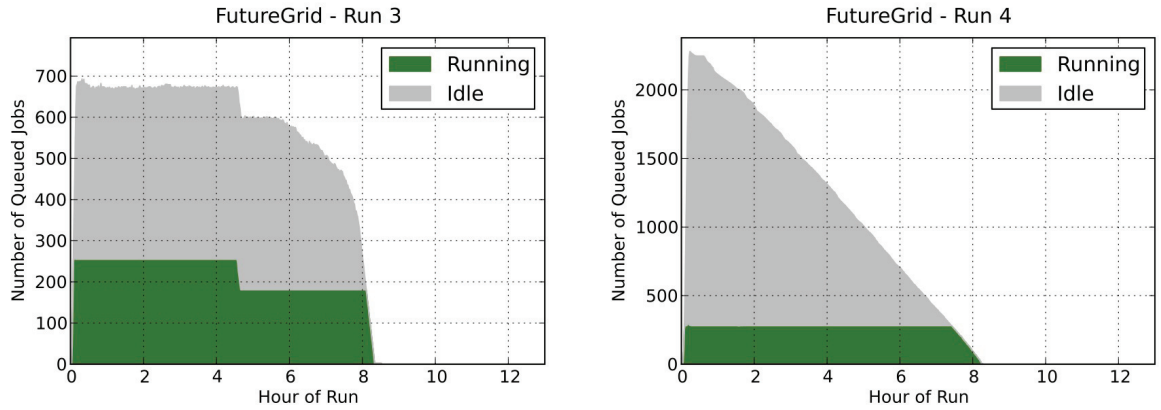

Figure 4: Periodogram runs on FutureGrid (Runs 1 and 2 omitted).

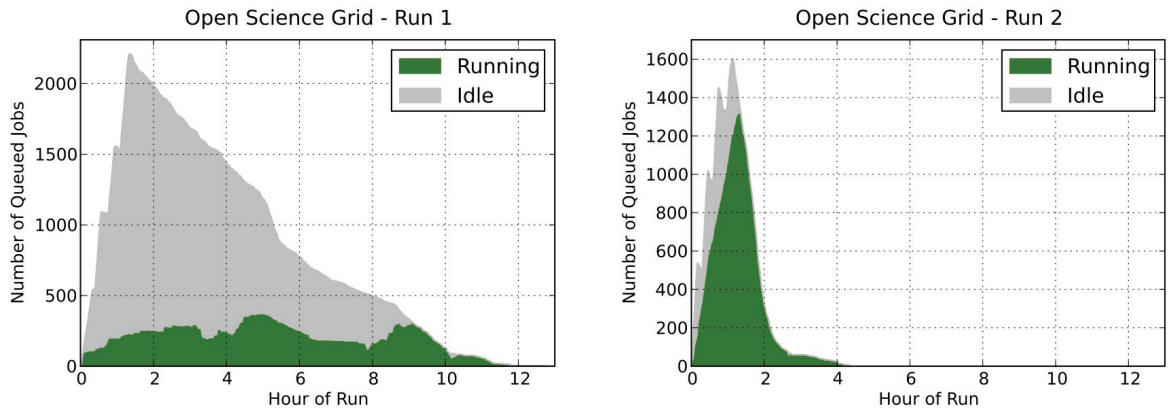

Figure 5: Periodograms executing on Open Science Grid (Run 3 omitted).

5.1. Runtime Comparison

Figure 3 shows the runs from Amazon EC2. We provisioned 256 cores using 32 spot instances. The shapes of the curves in the two graphs are very comparable. However, at the bottom of Figure 3 (left), there is a low tail of running jobs between hours 7 and 9. This tail was caused by a few long-running jobs that were not prioritized to start early enough in the workflow. For the run shown in Figure 3 (right), and subsequent runs, these tasks were scheduled at the start of the workflow to eliminate the long tail. Thus the workflow in Figure 3 (right) is more representative of the typical behavior on Amazon EC2. In general, the runtime behavior and performance on Amazon EC2 is very regular and predictable due to the stability and homogeneity of the resources. We encountered no failures when running the workflow on EC2, and we were able to fully utilize all of the resources we requested.

Figure 4 shows the experiments on FutureGrid. The first experiment in Figure 4 (left) shows a run where the workflow system was configured to limit the number of queued idle jobs to 420. Thus, the triangular peak that can be seen in all other workflow diagram has been flattened out. This restriction was removed for the repeat experiment in Figure 4 (right). Regardless of the job throttling, both workflows ran for slightly above eight hours.

In Figure 4 (left), a sudden decrease in resources occurred during the experiment when FutureGrid's Sierra cluster experienced a power outage. The number of idle jobs continued to be throttled to the same value, but number of available resources dropped. Our experience on FutureGrid, and with other academic clouds, suggests that these types of failures are relatively common. As academic clouds mature, we would expect these failures to decrease in number.

Comparing Amazon EC2 with FutureGrid using Figure 3 (right) and Figure 4 (right) suggests that performance is comparable on the two infrastructures. Both experiments provisioned the same number of remote resources, and the

Digital Object Indentifier 10.1109/MCSE.2013.44 1070-9924/$26.00 2013 IEEE

shape of the graphs is similar (primarily as a result of the configuration of the workflow management system). While the ramp-up for Amazon EC2 is slightly slower compared to FutureGrid, the Amazon experiments typically finish an hour earlier. However, this is likely due to differences in the performance of the network and server hardware rather than fundamental differences between the two platforms.

Figure 5 shows the jobs over time graphs for the experiments on OSG. Due to the highly variable resource availability on OSG the performance varied significantly, with workflow execution times between 4 and 12 hours, and as little as 300 or as many as 1,300 resources used at one time. The ramp-up of job release is similar to the other experiments, yet not as smooth. The less steep slope and the spikes seen in Figure 5 are present because the OSG submit node only allows a small number of local Condor jobs in parallel, which means the sub-workflows are planned in batches and then submitted for execution. Each spike indicates such a batch. Depending on the number of idle jobs, and the low-priority resources available to the VO, resources are dynamically provisioned based on the needs of the workflow, and de-provisioned when high-priority jobs crowd out workflow jobs. Figure 5 (left) shows one extreme, where only a few hundred resources were available. Figure 5 (right) shows the opposite, where a large number of resources were allocated in a short period of time.

5.2. Discussion

The main advantages of EC2 for the periodogram workflow were its scale and stability. While it is possible for Amazon to run out of available resources, such an event is rare in our experience and we were able to provision all the resources we required with no trouble. In addition, we experienced no failures while using EC2. The main drawback of commercial clouds like EC2 when compared with academic systems is the cost: academic systems are essentially free to the user, while commercial clouds can have a significant cost for large, distributed science applications. The costs of using EC2 depend on the amount of computation, storage, and data transfer required. In the case of the periodogram workflow, which is small in comparison to many scientific workflows, the costs were relatively modest, totaling about $65 for each run. We were able to reduce costs by compressing data to minimize transfer costs, and by using spot instances to decrease the cost of VM instances. EC2 spot instances are particularly useful for workflow applications because they reduce costs without any significant downside because workflow systems are designed to recover automatically in the case that a worker disappears, such as when a spot instance is terminated by EC2.

EC2 and FutureGrid provide very similar platforms in terms of technical features, and they both require a substantial amount of system administration to setup and maintain VM images and to deploy applications. FutureGrid provides a significant number of resources to explore cloud computing in a scientific context, however, the total number of resources available to the average user is more limited than EC2 or OSG, requiring the use of multiple clusters within FutureGrid for larger experiments. Our application experienced more failures on FutureGrid than the other infrastructures, but we attribute that to its status as a testbed for cloud computing rather than a production resource for science applications.

Open Science Grid provides a computational resource to a variety of users that is similar to the cloud infrastructures, but the challenges of the two are somewhat different. In the cloud case, the user is able to obtain resources relatively easily, but is required to set up and maintain virtual machines, which can be a significant burden. On OSG the compute nodes are already set up and maintained, but users' jobs might have lower priority and face preemption that makes it difficult to acquire sufficient resources. When resources are plentiful, workflows running on OSG can have excellent performance. And while failures were relatively common on OSG, they were typically isolated to a single worker and our workflow management software was able to mask all of them.

6. Related Work

Several previous authors have described the potential opportunities and challenges of running workflows in the cloud [17,42]. Many of the benefits of cloud workflows identified by these authors, such as easy access to resources and

Digital Object Indentifier 10.1109/MCSE.2013.44 1070-9924/$26.00 2013 IEEE

elasticity, as well as the challenges, such as system administration, complexity and data management, are consistent with our practical experiences in deploying real workflows in the cloud.

In recent years, several researchers have been developing new workflow systems that are customized for clouds [10,23,27,41] or adding cloud capabilities to existing workflow systems [37]. These efforts are important because they will enable more efficient execution of workflow applications on cloud platforms. In our experience we have found that an existing workflow system developed primarily for clusters and grids (Pegasus) works just as well, if not better, on clouds. The real challenge has been developing tools and techniques for deploying existing solutions in the cloud.

Recently there has been a large amount of effort focused on developing new algorithms for workflows to take advantage of the unique pricing model and elasticity of infrastructure clouds. For example, many researchers have investigated ways to take advantage of cloud elasticity to develop dynamic provisioning algorithms for workflows [16,18,22,28,31]. Other researchers have developed new cloud workflow scheduling algorithms that optimize for cost, performance, and other QoS metrics [2,26,30,35]. Finally, many researchers have been investigating techniques for deploying data-intensive workflows in the cloud using unique architectures that are difficult to deploy on traditional platforms such as grids [3,38–40].

In our own previous work we have studied the cost and performance of workflows in the cloud via simulation [7], using an experimental Nimbus cloud [12], using individual EC2 nodes [15], and using a variety of different intermediate storage systems on EC2 [14]. Vecchiola, et al. have done similar investigations on EC2 and Grid5000 [36]. These studies primarily analyze the performance and cost of workflows in the cloud, rather than the practical experience of deploying workflows in the cloud. In this study we relate the practical experience of trying to run a non-trivial scientific workflow application on three different infrastructures and compare the benefits and challenges of each platform.

7. Conclusions and Future Work

In this paper we described our experiences deploying a typical scientific workflow application across three different infrastructures: a grid (Open Science Grid), a commercial cloud (Amazon EC2), and an academic cloud (FutureGrid). Our goal was to compare these infrastructures in a way that helps scientists understand the benefits and drawbacks of each infrastructure, and to help them make informed decisions about where to deploy their own applications.

In comparing the three infrastructures we found that:

- Getting started on the grid and academic cloud took significantly more time than getting started on the commercial cloud. The former have relatively long processes for obtaining accounts and setting up the necessary tooling to begin running applications, while the latter has a simple online form.
- All the infrastructures had steep learning curves. Grid security configuration and tooling was very difficult to set up, and the grid suffered from errors that are difficult to debug. The clouds were somewhat easier to get started with, but maintaining VM images was very tedious and the system administration skills required may be beyond the capabilities of many users.
- Grids and academic clouds were more attractive from a cost perspective than commercial clouds because they are effectively free to the user. At the same time, the cost of the commercial cloud was not exorbitant and we were able to use several techniques to reduce costs.
- Acquiring resources from the commercial cloud infrastructure was easier than the grid or academic cloud. The policies of the grid infrastructure made it occasionally difficult to acquire resources and the academic cloud was simply too small to easily support our application.
- Regardless of the infrastructure, we found it helpful to use provisioning tools such as Wrangler and GlideinWMS to assist us in deploying our application.
- Failures were more common on grids and academic clouds than commercial clouds.

Digital Object Indentifier 10.1109/MCSE.2013.44 1070-9924/$26.00 2013 IEEE

- The clouds were more predictable than the grid and gave more uniform performance. On-demand provisioning in clouds either succeeds or fails immediately. In comparison, user jobs may wait indefinitely in a grid queue without any indication of when, if ever, they may start.

In the future we will continue to investigate a number of issues related to deploying scientific workflows in the cloud, including:

- Investigating data management solutions such as alternative protocols and storage solutions. The cloud may enable new storage architectures to be deployed that are designed specifically for workflows.
- Continuing to develop new techniques for deploying workflows in the cloud [13] and for running repeatable experiments [29].
- Studying new algorithms for dynamic provisioning across grids and clouds. Provisioning tools are needed for clouds similar to what is available on the grid, such as GlideinWMS.
- Developing hosted workflow management systems that will simplify workflow development by eliminating the burden of installing and maintaining complex middleware.

This research will aid in the development of capabilities for science applications in the cloud that cannot be provided by traditional scientific computing platforms such as grids.

8. Acknowledgements

This material is based upon work supported in part by the National Science Foundation under Grants (OCI-0943725, 0910812).

This research was done using resources provided by the Open Science Grid, which is supported by the NSF and the DOE's Office of Science.

The use of Amazon EC2 resources was supported by an AWS in Education research grant.

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).

9. References

[1] Amazon Elastic Compute Cloud (EC2), http://aws.amazon.com/ec2.
[2] K. Bessai et al., "Bi-criteria Workflow Tasks Allocation and Scheduling in Cloud Computing Environments," *2012 IEEE 5th International Conference on Cloud Computing (CLOUD)*, 2012.
[3] Ü.V. Çatalyürek, K. Kaya, and B. Uçar, "Integrated data placement and task assignment for scientific workflows in clouds," *Proceedings of the fourth international workshop on Data-intensive distributed computing*, 2011.
[4] Committee for a Decadal Survey of Astronomy and Astrophysics, "New Worlds, New Horizons in Astronomy and Astrophysics," National Research Council. 2010.
[5] Cycle Computing, New CycleCloud HPC Cluster Is a Triple Threat: 30000 cores, $1279/Hour, & Grill monitoring GUI for Chef, http://blog.cyclecomputing.com/2011/09/new-cyclecloud-cluster-is-a-triple-threat-30000-cores-massive-spot-instances-grill-chef-monitoring-g.html.
[6] E. Deelman et al., "Pegasus: A framework for mapping complex scientific workflows onto distributed systems," *Scientific Programming*, vol. 13, no. 3, pp. 219–237, 2005.
[7] E. Deelman et al., "The Cost of Doing Science on the Cloud: The Montage Example," *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, 2008.
[8] Extreme Science and Engineering Discovery Environment (XSEDE), http://www.xsede.org.
[9] I. Foster, C. Kesselman, and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," *International Journal of High Performance Computing Applications*, vol. 15, no. 3, pp. 200–222, 2001.

Digital Object Indentifier 10.1109/MCSE.2013.44 1070-9924/$26.00 2013 IEEE

[10] D. Franz et al., "A Workflow Engine for Computing Clouds," *The Second International Conference on Cloud Computing, GRIDs, and Virtualization*, 2011.
[11] FutureGrid, http://futuregrid.org/.
[12] C. Hoffa et al., "On the Use of Cloud Computing for Scientific Workflows," *3rd International Workshop on Scientific Workflows and Business Workflow Standards in e-Science (SWBES '08)*, 2008.
[13] G. Juve and E. Deelman, "Automating Application Deployment in Infrastructure Clouds," *3rd IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, 2011.
[14] G. Juve et al., "Data Sharing Options for Scientific Workflows on Amazon EC2," *2010 ACM/IEEE conference on Supercomputing (SC 10)*, 2010.
[15] G. Juve et al., "Scientific workflow applications on Amazon EC2," *5th IEEE International Conference on E-Science Workshops*, 2009.
[16] C. Lin and S. Lu, "Scheduling Scientific Workflows Elastically for Cloud Computing," *2011 IEEE International Conference on Cloud Computing (CLOUD)*, 2011.
[17] X. Liu et al., "Workflow Systems in the Cloud," *The Design of Cloud Workflow Systems*, New York, NY, USA: Springer, 2012, pp. 1–11.
[18] M. Mao and M. Humphrey, "Auto-scaling to minimize cost and meet application deadlines in cloud workflows," *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis (SC 11)*, 2011.
[19] NASA, Kepler, http://kepler.nasa.gov/.
[20] Nimbus, http://www.nimbusproject.org.
[21] D. Nurmi et al., "The Eucalyptus Open-source Cloud-computing System," *9th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 09)*, 2009.
[22] D. de Oliveira et al., "An adaptive parallel execution strategy for cloud-based scientific workflows," *Concurrency and Computation: Practice and Experience*, vol. 24, no. 13, pp. 1531–1550, 2012.
[23] D. de Oliveira et al., "SciCumulus: A Lightweight Cloud Middleware to Explore Many Task Computing Paradigm in Scientific Workflows," *2010 IEEE 3rd International Conference on Cloud Computing*, 2010.
[24] Open Science Grid, http://www.opensciencegrid.org.
[25] OpenStack, http://openstack.org.
[26] S. Ostermann and R. Prodan, "Impact of Variable Priced Cloud Resources on Scientific Workflow Scheduling," *Euro-Par 2012 Parallel Processing*, C. Kaklamanis, T. Papatheodorou, and P. Spirakis, eds., Berlin/Heidelberg: Springer, 2012, pp. 350–362.
[27] S. Pandey, D. Karunamoorthy, and R. Buyya, "Workflow Engine for Clouds," *Cloud Computing*, R. Buyya, J. Broberg, and A. Goscinski, eds., John Wiley & Sons, Inc., 2011, pp. 321–344.
[28] G. Papuzzo and G. Spezzano, "Autonomic management of workflows on hybrid Grid-Cloud infrastructure," *2011 7th International Conference on Network and Service Management (CNSM)*, 2011.
[29] precip - Pegasus Repeatable Experiments for the Cloud in Python, http://pegasus.isi.edu/projects/precip.
[30] L. Ramakrishnan et al., "Deadline-sensitive workflow orchestration without explicit resource control," *Journal of Parallel and Distributed Computing*, vol. 71, no. 3, pp. 343–353, Mar. 2011.
[31] C.J. Reynolds et al., "Scientific Workflow Makespan Reduction through Cloud Augmented Desktop Grids," *2011 IEEE Third International Conference on Cloud Computing Technology and Science (CloudCom)*, 2011.
[32] I. Sfiligoi, "glideinWMS—a generic pilot-based workload management system," *Journal of Physics: Conference Series*, vol. 119, no. 6, p. 062044, 2008.
[33] The Large Synoptic Survey Telescope, http://www.lsst.org/lsst.
[34] The Rackspace Cloud, http://www.rackspace.com/cloud.
[35] R. Tolosana-Calasanz et al., "Enforcing QoS in scientific workflow systems enacted over Cloud infrastructures," *Journal of Computer and System Sciences*, vol. 78, no. 5, pp. 1300–1315, Sep. 2012.
[36] C. Vecchiola, S. Pandey, and R. Buyya, "High-Performance Cloud Computing: A View of Scientific Applications," *International Symposium on Parallel Architectures, Algorithms, and Networks*, 2009.
[37] J. Wang and I. Altintas, "Early Cloud Experiences with the Kepler Scientific Workflow System," *International Conference on Computational Science (ICCS)*, 2012.
[38] J. Wang, P. Korambath, and I. Altintas, "A Physical and Virtual Compute Cluster Resource Load Balancing Approach to Data-Parallel Scientific Workflow Scheduling," *2011 IEEE World Congress on Services (SERVICES)*, 2011.
[39] D. Yuan et al., "A data placement strategy in scientific cloud workflows," *Future Generation Computer Systems*, vol. 26, no. 8, pp. 1200–1214, Oct. 2010.
[40] D. Yuan et al., "On-demand minimum cost benchmarking for intermediate dataset storage in scientific cloud workflow systems," *Journal of Parallel and Distributed Computing*, vol. 71, no. 2, pp. 316–332, Feb. 2011.

Digital Object Indentifier 10.1109/MCSE.2013.44 1070-9924/$26.00 2013 IEEE

[41] Y. Zhao et al., "Designing and Deploying a Scientific Computing Cloud Platform," *2012 ACM/IEEE 13th International Conference on Grid Computing (GRID)*, 2012.
[42] Y. Zhao et al., "Opportunities and Challenges in Running Scientific Workflows on the Cloud," *2011 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, 2011.

Gideon Juve is a Computer Scientist at USC/ISI (PhD in Computer Science, USC in 2012). His research focuses on enabling and optimizing scientific workflows on clusters, grids and clouds.

Ewa Deelman is a Project Leader at USC/ISI (PhD in Computer Science, RPI in 1997). Her research interests include distributed scientific environments, with emphasis on workflow management.

Bruce Berriman is a Senior Scientist at IPAC, Caltech (PhD in Astronomy, Caltech in 1983). His research centers on investigating the applicability of emerging technologies to astronomy.

Mats Rynge is a Research Programmer at USC/ISI (BS in Computer Science, UCLA in 2002). His research interests include distributed and high performance computing.

Jens Vöckler received his MS in Electric Engineering from Leibniz Universität Hannover, in 1997.

Digital Object Indentifier 10.1109/MCSE.2013.44 1070-9924/$26.00 2013 IEEE