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

Scientists have several computing infrastructures available to conduct their research, including grids and public or private clouds. The authors explore the use of these cyberinfrastructures to execute scientific workflows, an important class of scientific applications.

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,[1] which recommends national priorities for the field, declared the time domain "the last frontier" in astrophysics. These surveys are already beginning to produce datasets too large to be analyzed by local resources, and will culminate near the end of this decade with the Large Synoptic Survey Telescope (LSST; www.lsst.org/lsst), which expects to produce petabyte-scale datasets 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. Consequently, workflows enable scientists to easily define multistage 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's important to understand the benefits and drawbacks of the various cyberinfrastructures available, so that researchers 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 (OSG; www.opensciencegrid.org) or XSEDE (www.xsede.org), commercial clouds such as the Amazon Elastic Compute Cloud (EC2; http://aws.amazon.com/ec2) or the Rackspace Cloud (www.rackspace.com/cloud), and academic clouds such as FutureGrid (http://futuregrid.org). 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

1521-9615/13/$31.00 © 2013 IEEE
Copublished by the IEEE CS and the AIP

Gideon Juve, Mats Rynge, Ewa Deelman, and Jens-S. Vöckler
University of Southern California
G. Bruce Berriman
California Institute of Technology

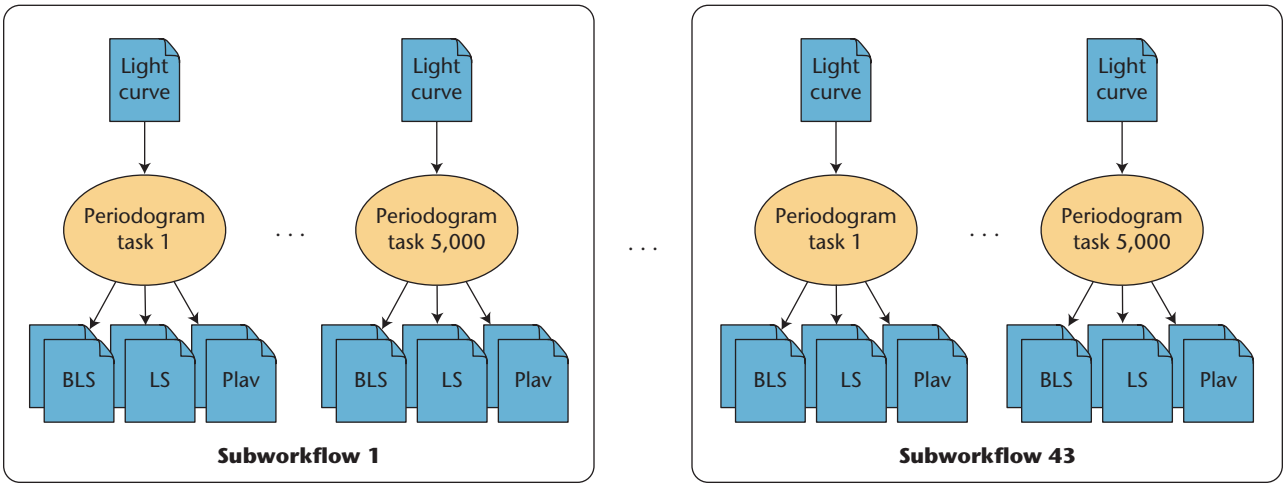

Figure 1. Kepler periodogram workflow. Its 43 subworkflows contain approximately 5,000 tasks each. LS = Lomb Scargle, BLS = Box Least Squares, and Plav = Plavchan Binless Phase Dispersion. These are three common algorithms used in periodogram calculations; they differ in their assumptions about the shape of underlying periodicities present in time-series datasets.

other traditional high-performance computing (HPC) systems, which provide only best-effort service and are configured and maintained by resource 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 trade-offs in terms of effort, cost, and performance. In this article, we describe some of the opportunities and challenges of running scientific workflows on grids as well as in commercial and academic clouds. We chose three execution infrastructures that are representative of each category: the OSG, Amazon EC2, and FutureGrid. We describe our experiences using these infrastructures for running a scientific workflow application that analyzes data from NASA's Kepler mission (http://kepler.nasa.gov). We also compare the infrastructures across several dimensions, including obtaining an account, environment setup, cost, provisioning, and resource availability. (See the "Related Work in Cloud Workflows" sidebar for more background.)

Periodograms: An Astronomy Application

Our example workflow application processes time-series data collected by NASA's Kepler mission. The Kepler satellite uses high-precision photometry to search for exoplanets transiting their host stars. In 2009, the Kepler mission began a multiyear 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 a star's brightness 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.

To support the analysis of Kepler's 210,664 light-curve dataset, we developed a workflow using the Pegasus workflow management system.[2] The workflow, illustrated in Figure 1, consists of 43 subworkflows, 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. From estimates of the tasks' runtimes, 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 Gbytes of input data, and produces 105 Gbytes of output data in 1.2 million output files.

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), which maintains a queue of jobs that are ready to run on remote resources. The workflow is planned by Pegasus and submitted to

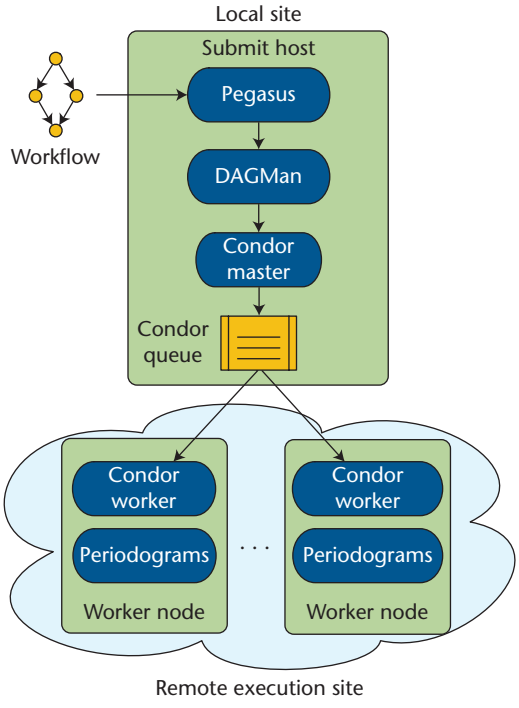

Figure 2. Overview of the execution environment. The workflow management system resides on a machine controlled by the workflow developer called the submit host (SH), which maintains a queue of jobs that are ready to run on remote resources.

DAGMan, a workflow engine that releases jobs in the correct order to Condor, a job scheduling system, for execution on remote worker nodes. In our configuration, the SH was located at the Information Sciences Institute (ISI) in Marina del Rey, California, for the cloud experiments, and at the Renaissance Computing Institute (RENCI) in North Carolina for the OSG experiments. For each job, the input data was transferred automatically from the SH to the remote workers, and the output data was transferred directly from the remote workers back to the SH. 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 percent.

Grid: Open Science Grid

The Open Science Grid (OSG) is a distributed computing consortium of major science experiments and academic institutions; approximately 80 sites today offer compute and storage resources to OSG users. OSG is structured around virtual organizations (VOs),[3] 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 article, the *Engagement VO*, helps new users get started and doesn't own any computational resources; instead, its 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, enabling their jobs to preempt opportunistic jobs.

Commercial Cloud: Amazon EC2

Amazon Elastic Cloud2 (EC2), a large commercial cloud operated by Amazon.com, is part of a suite of cloud services called Amazon Web Services (AWS). AWS provides computational, storage, and communication infrastructure on demand via Web-based APIs. EC2 is a service for provisioning virtual machine (VM) instances from Amazon's data centers. It lets users deploy VM images with customized operating systems, libraries, and application code on VMs with a variety of predefined hardware configurations (CPU, memory, disk) and prices. EC2 users are only billed for the resources they use; there are charges associated with VM instance hours and data transfers to and from Amazon's datacenter.

Academic Cloud: FutureGrid

The FutureGrid project 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 (www.nimbusproject.org), Eucalyptus,[4] and OpenStack (http://openstack.org), as well as "bare metal" systems for experiments aimed at minimizing overhead and maximizing performance.

Infrastructure Setup

Getting access to these three infrastructures involves following a variety of steps, which we describe next.

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 one. Project proposal turnaround time is less than a week—often, just one business day. With an account and an approved project, a user can upload his or her public Secure Shell (SSH) key through the website and gain 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, for example, he or she must first request an X.509 user certificate from DOEGrids. Once the certificate request has been verified and the certificate issued, the user can request membership in the Engagement VO by using the VOMS (VO Membership Service) to request an account on the community's SH. The entire process typically takes about a week.

Environment Setup

The procedure for creating the necessary execution environment on cloud infrastructures (especially Amazon EC2 and FutureGrid) is very similar. The user must create an image that contains Condor, Pegasus, and any necessary application software and data. Installing these packages in the image reduces the execution environment's setup time and reduces the amount of data that needs 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 weeks if the user isn't 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 aren't 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 preconfigured to execute workflows. For many scientists, this approach is more attractive than a cloud deployment because the VO installs and maintains the infrastructure software, which can significantly reduce the startup time. For the experiments in this article, we used an SH 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 instances will look the same, OSG resources are much more heterogeneous, and each resource can 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 is prestaged to the resources, so the workflow has to stage those executables for each job.

Resource Provisioning

Given the complexity of the execution environments required to support distributed applications such as workflows, it's often necessary to use tools that support resource provisioning and configuration. These tools help 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.

We provisioned the resources on Amazon EC2 with Wrangler,[5] 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 resource provisioning and software and service installation and configuration. Specifically, it supports plugins that enable users to define custom behaviors for their application, allowing dependencies to be specified between nodes. Users can create complex deployments by composing plugins to install and configure services on several interdependent nodes.

We manually provisioned the resources on FutureGrid from the Nimbus 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.

We used GlideinWMS[6] to provision resources on OSG. It has a distributed design with two

Related Work in Cloud Workflows

Several previous works have described the potential opportunities and challenges of running workflows in the cloud.[1,2] Many of the benefits of cloud workflows identified by these authors, such as easy access to resources and 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 developed new workflow systems customized for clouds[3–6] or added cloud capabilities to existing workflow systems.[7] These efforts are important because they enable more efficient execution of workflow applications on cloud platforms. In our experience, we've 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, a large amount of effort has 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.[8–12] Other researchers have developed new cloud workflow-scheduling algorithms that optimize for cost, performance, and other quality-of-service metrics.[13–16] 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.[17–20]

In our own previous work, we've studied the cost and performance of workflows in the cloud via simulation,[21] using an experimental Nimbus cloud,[22] individual Elastic Compute Cloud (EC2) nodes,[23] and a variety of different intermediate storage systems on EC2.[24] Other researchers have done similar investigations on EC2 and Grid5000.[25] 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 our study, we relate the practical experience of trying to run a nontrivial scientific workflow application on three different infrastructures and compare the benefits and challenges of each platform.

References

1. X. Liu et al., "Workflow Systems in the Cloud," *The Design of Cloud Workflow Systems*, Springer, 2012, pp. 1–11.
2. Y. Zhao et al., "Opportunities and Challenges in Running Scientific Workflows on the Cloud," *Proc. Int'l Conf. Cyber-Enabled Distributed Computing and Knowledge Discovery* (CyberC), IEEE CS, 2011, pp. 455–462.
3. D. Franz et al., "A Workflow Engine for Computing Clouds," *Proc. 2nd Int'l Conf. Cloud Computing, GRIDs, and Virtualization*, Int'l Academy, Research, and Industry Assoc., 2011, pp 1–6.
4. D. de Oliveira et al., "SciCumulus: A Lightweight Cloud Middleware to Explore Many Task Computing Paradigm in Scientific Workflows," *Proc. IEEE 3rd Int'l Conf. Cloud Computing*, IEEE, 2010, pp. 378–385.
5. S. Pandey, D. Karunamoorthy, and R. Buyya, "Workflow Engine for Clouds," *Cloud Computing*, R. Buyya, J. Broberg, and A. Goscinski, eds., John Wiley & Sons, 2011, pp. 321–344.
6. Y. Zhao et al., "Designing and Deploying a Scientific Computing Cloud Platform," *Proc. ACM/IEEE 13th Int'l Conf. Grid Computing* (GRID), IEEE CS, 2012, pp. 104–113.

main components: a *front end* for each SH and a *factory* for each compute resource. The front end queries the Condor job queue to determine when the resource pool needs to grow. If an increase is desired, it sends a request to the factory to provision more resources by submitting pilot jobs. For our OSG experiments, we installed the front end on the Engagement VO SH at RENCI and used a community factory at the University of California, San Diego (UCSD), for provisioning. Once pilot jobs start up on a compute resource, they register back to the SH at RENCI, which uses them to execute queued workflow jobs.

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 contrast, FutureGrid and OSG operate under an allocation model, common in academic computing, where users apply for access to computing resources and are allocated those resources based on their proposal's scientific merit. From a scientist's perspective, academic resources such as FutureGrid and OSG are more attractive options for science applications than commercial clouds such as EC2 because they're effectively free.

Amazon has a very complex pricing model for EC2 and related services. In general, it charges per hour, gigabyte, and operation across all their services. For the experiments in this article, we were charged for two services: data transfer and VM instances. Data transfer from our SH to EC2 was free, but data transfer from EC2 back to our SH was billed at $0.12/Gbyte, or $2.44/workflow for the 20 Gbytes of compressed output data (105 Gbytes uncompressed) produced by the periodogram workflow. Amazon charges

7. J. Wang and I. Altintas, "Early Cloud Experiences with the Kepler Scientific Workflow System," *Proc. Int'l Conf. Computational Science* (ICCS), Elsevier, 2012, pp. 1630–1634; doi:10.1016/j.procs.2012.04.179.
8. C. Lin and S. Lu, "Scheduling Scientific Workflows Elastically for Cloud Computing," *Proc. IEEE Int'l Conf. Cloud Computing* (CLOUD), IEEE, 2011, pp. 746–747; doi:10.1109/CLOUD.2011.110.
9. M. Mao and M. Humphrey, "Auto-Scaling to Minimize Cost and Meet Application Deadlines in Cloud Workflows," *Proc. of 2011 Int'l Conf. High Performance Computing, Networking, Storage and Analysis* (SC 11), ACM, 2011, article no. 49; doi:10.1145/2063384.2063449.
10. 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, 2012, pp. 1531–1550.
11. G. Papuzzo and G. Spezzano, "Autonomic Management of Workflows on Hybrid Grid-Cloud Infrastructure," *Proc. 7th Int'l Conf. Network and Service Management* (CNSM), Int'l Federation for Information Processing, 2011, pp. 230–233.
12. C.J. Reynolds et al., "Scientific Workflow Makespan Reduction through Cloud Augmented Desktop Grids," *Proc. IEEE 3rd Int'l Conf. Cloud Computing Technology and Science* (CloudCom), IEEE, 2011, pp. 18–23; doi:10.1109/CloudCom.2011.13.
13. K. Bessai et al., "Bi-Criteria Workflow Tasks Allocation and Scheduling in Cloud Computing Environments," *Proc. IEEE 5th Int'l Conf. Cloud Computing* (CLOUD), IEEE, 2012, pp. 638–645; doi:10.1109/CLOUD.2012.83.
14. 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., Springer, 2012, pp. 350–362.
15. L. Ramakrishnan et al., "Deadline-Sensitive Workflow Orchestration without Explicit Resource Control," *J. Parallel and Distributed Computing*, vol. 71, no. 3, 2011, pp. 343–353.
16. R. Tolosana-Calasanz et al., "Enforcing QoS in Scientific Workflow Systems Enacted Over Cloud Infrastructures," *J. Computer and System Sciences*, vol. 78, no. 5, 2012, pp. 1300–1315.
17. Ü.V. Çatalyürek, K. Kaya, and B. Uçar, "Integrated Data Placement and Task Assignment for Scientific Workflows In Clouds," *Proc. 4th Int'l Workshop Data-Intensive Distributed Computing*, ACM, 2011, pp. 45–54.
18. J. Wang, P. Korambath, and I. Altintas, "A Physical and Virtual Compute Cluster Resource Load Balancing Approach to Data-Parallel Scientific Workflow Scheduling," *Proc. IEEE World Congress on Services*, IEEE, 2011, pp. 212–215; doi:10.1109/SERVICES.2011.50.
19. D. Yuan et al., "A Data Placement Strategy in Scientific Cloud Workflows," *Future Generation Computer Systems*, vol. 26, no. 8, 2010, pp. 1200–1214.
20. D. Yuan et al., "On-Demand Minimum Cost Benchmarking for Intermediate Dataset Storage in Scientific Cloud Workflow Systems," *J. Parallel and Distributed Computing*, vol. 71, no. 2, 2011, pp. 316–332.
21. E. Deelman et al., "The Cost of Doing Science on the Cloud: The Montage Example," *Proc. 2008 ACM/IEEE Conf. Supercomputing*, IEEE, 2008, article no. 50.
22. C. Hoffa et al., "On the Use of Cloud Computing for Scientific Workflows," *Proc. 3rd Int'l Workshop Scientific Workflows and Business Workflow Standards in e-Science*, IEEE CS, 2008, pp. 640–645.
23. G. Juve et al., "Scientific Workflow Applications on Amazon EC2," *Proc. 5th IEEE Int'l Conf. E-Science Workshops*, IEEE, 2009, pp. 59–66; doi:10.1109/ESCIW.2009.5408002.
24. G. Juve et al., "Data Sharing Options for Scientific Workflows on Amazon EC2," *Proc. 2010 ACM/IEEE Conf. Supercomputing* (SC 10), IEEE, 2010, pp. 1–9, doi:10.1109/SC.2010.17.
25. C. Vecchiola, S. Pandey, and R. Buyya, "High-Performance Cloud Computing: A View of Scientific Applications," *Proc. Int'l Symp. Parallel Architectures, Algorithms, and Networks*, IEEE, 2009, pp. 4–16; doi:10.1109/I-SPAN.2009.150.

for EC2 VM instances by the hour and has two types of instance requests: on-demand and spot. Users of spot instances bid the maximum amount they're 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 current bids using a proprietary algorithm. Typically, spot prices are significantly lower than on-demand prices. For the experiments in this article, we used spot instances for which we were charged an average of $0.24 per hour (a savings of 65 percent). The total cost of spot instances for the periodogram application was approximately $61.44 per workflow (32 instances × 8 hours).

Resource Availability

Although the size of EC2 isn't public information, it's 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 (http://blog.cyclecomputing.com/2011/09/new-cyclecloud-cluster-is-a-triple-threat-30000-cores-massive-spot-instances-grill-chef-monitoring-g.html). In our experience, we've always been able to provision all the resources we required for an application from EC2. For the experiments in this article, 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.

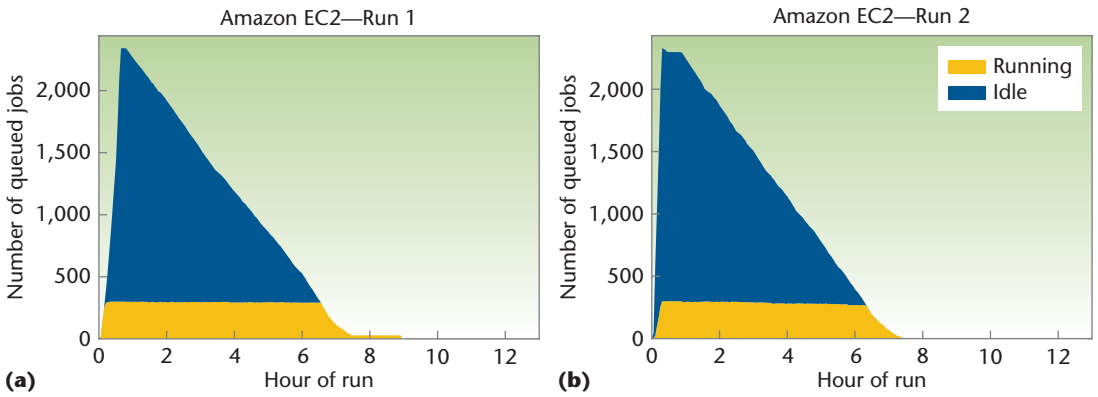

Figure 3. Periodogram runs on Amazon EC2 (run 3 omitted). (a) Run 1 and (b) run 2.

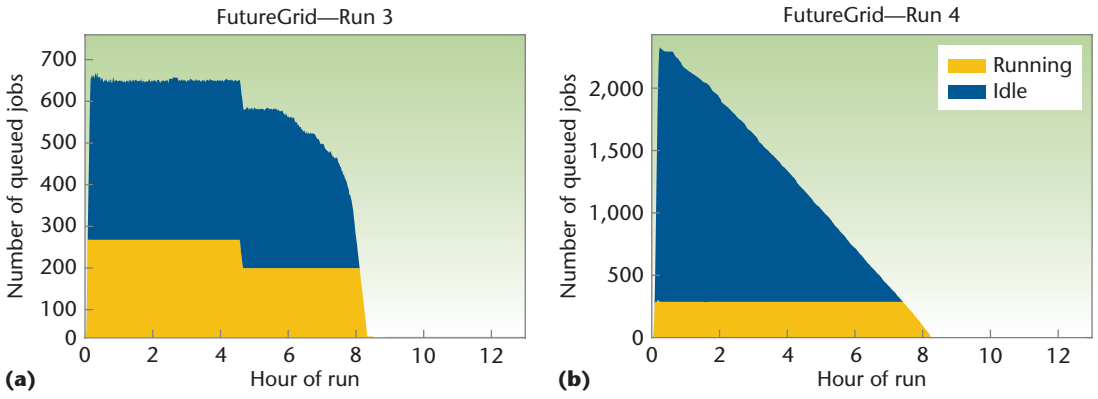

Figure 4. Periodogram runs on FutureGrid (runs 1 and 2 omitted). (a) Run 3 and (b) run 4.

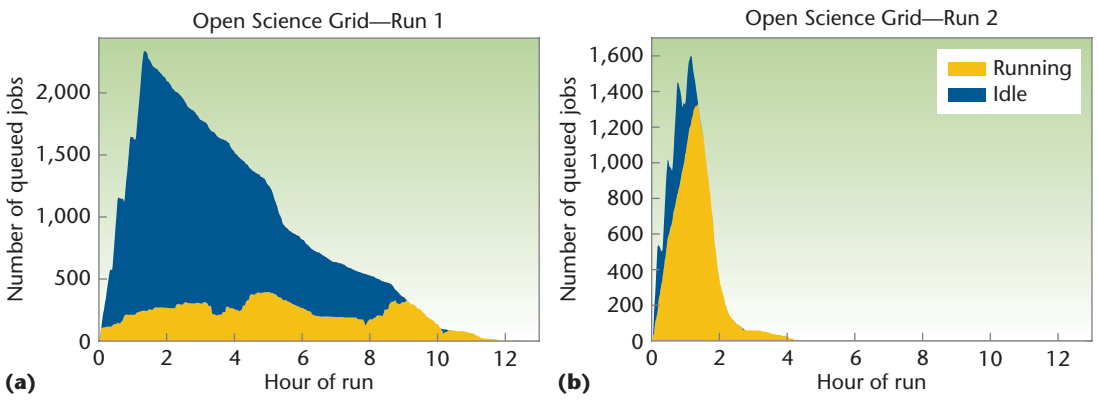

Figure 5. Periodograms executing on Open Science Grid (run 3 omitted). (a) Run 1 and (b) run 2.

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 might have a low priority or be preempted if the resource owner submits more jobs. Most of the time, resource polices and system status aren't available to the user, which means that opportunistic users won't 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.

Experiments

We ran the workflow several times on each infrastructure 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 yellow part of each figure shows the number of concurrently running jobs; the blue curve, stacked on top of the yellow curve, shows the number of jobs in the Condor queue waiting for a resource to become idle.

Runtime Comparison

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

Figure 4 shows the experiments on FutureGrid. The first experiment in Figure 4a shows a run in which we configured the workflow system to limit the number of queued idle jobs to 420. Thus, the triangular peak that can be seen in all other workflow diagram has flattened out. We removed this restriction for the repeat experiment in Figure 4b. Regardless of job throttling, both workflows ran for slightly longer than eight hours.

In Figure 4a, 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 the 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 expect that these failures will decrease in number.

Comparing Amazon EC2 with FutureGrid using Figures 3b and 4b suggests that performance is comparable on the two infrastructures. We provisioned both experiments with the same number of remote resources, and the graph shapes are similar (primarily as a result of the workflow management system's configuration). 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 network and server hardware performance rather than fundamental differences between the two platforms.

Figure 5 shows the jobs over time 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 subworkflows 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 workflow needs and de-provisioned when high-priority jobs crowd out workflow jobs. Figure 5a shows one extreme, where only a few hundred resources were available; Figure 5b shows the opposite, with a large number of resources allocated in a short period of time.

Discussion

The main advantages of EC2 for the periodogram workflow were its scale and stability. Although it's 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 such as EC2 when compared with academic systems is the cost: academic systems are essentially free to the user, whereas 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, since workflow systems are designed to recover automatically when resource availability changes (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, but 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.

OSG provides a computational resource to a variety of users that's similar to 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 compute node, and our workflow management software was able to mask all of them.

Our goal was to compare these infrastructures in a way that helps scientists understand each infrastructure's benefits and drawbacks, and to help them make informed decisions about where to deploy their own applications. We learned several things:

- Getting started on the grid and academic clouds 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 tedious, and the system administration skills required might 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.
- 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 might wait indefinitely in a grid queue without any indication of when, if ever, they can start.

In the future, we'll continue to investigate several 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[5] and for running repeatable experiments (http://pegasus.isi.edu/projects/precip);
- studying new algorithms for dynamic provisioning across grids and clouds (provisioning tools are needed for clouds similar to what's available on the grid, such as GlideinWMS); and
- 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 can't be provided by traditional scientific computing platforms, such as grids.

Acknowledgments

This material is based on work supported in part by the US 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 Amazon Web Services (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).

References

1. Committee for a Decadal Survey of Astronomy and Astrophysics, Nat'l Research Council, *New Worlds, New Horizons in Astronomy and Astrophysics*, Nat'l Academies Press, 2010.
2. E. Deelman et al., "Pegasus: A Framework for Mapping Complex Scientific Workflows onto Distributed Systems," *Scientific Programming*, vol. 13, no. 3, 2005, pp. 219–237.
3. I. Foster, C. Kesselman, and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," *Int'l J. High-Performance Computing Applications,* vol. 15, no. 3, 2001, pp. 200–222.
4. D. Nurmi et al., "The Eucalyptus Open-Source Cloud-Computing System," *Proc. 9th IEEE/ACM Int'l Symp. Cluster Computing and the Grid* (CCGrid 09), IEEE CS, 2009, pp. 124–131.
5. G. Juve and E. Deelman, "Automating Application Deployment in Infrastructure Clouds," *Proc. 3rd IEEE Int'l Conf. Cloud Computing Technology and Science* (CloudCom), IEEE, 2011, pp. 658–665; doi:10.1109/CloudCom.2011.102.
6. I. Sfiligoi, "GlideinWMS—A Generic Pilot-Based Workload Management System," *J. Physics: Conf. Series*, vol. 119, no. 6, 2008, p. 062044.

Gideon Juve *is a computer scientist at the University of Southern California's Information Sciences Institute. His research focuses on enabling and optimizing scientific workflows on clusters, grids, and clouds. Juve has a PhD in computer science from the University of Southern California. Contact him at gideon@isi.edu.*

Mats Rynge *is a computer scientist at the University of Southern California's Information Sciences Institute. His research interests include distributed and high-performance computing. Rynge has a BS in computer science from the University of California, Los Angeles. Contact him at rynge@isi.edu.*

Ewa Deelman *is a research associate professor at the University of Southern California's Computer Science Department. Her research interests include distributed scientific environments, with an emphasis on workflow management. Deelman has a PhD in computer science from Rensselaer Polytechnic Institute. Contact her at deelman@isi.edu.*

Jens-S. Vöckler *is a senior research engineer at the University of Southern California's Information Sciences Institute. His research focuses on distributed computing. Vöckler has an MS in electrical engineering from Leibniz Universität Hannover. Contact him at voeckler@isi.edu.*

G. Bruce Berriman *is a senior scientist at the California Institute of Technology. His research focuses on investigating the applicability of emerging technologies to astronomy. Berriman has a PhD in astronomy from the California Institute of Technology. Contact him at gbb@ipac.caltech.edu.*