International Journal of High Performance Computing Applications OnlineFirst, published on December 4, 2009 as doi:10.1177/1094342009356432

Grids and Clouds: Making Workflow Applications Work in Heterogeneous Distributed Environments

Ewa Deelman

USC INFORMATION SCIENCES INSTITUTE, MARINA DEL REY, CA 90292, USA
(DEELMAN@ISI.EDU)

Abstract

Scientific workflows are frequently being used to model complex phenomena, to analyze instrumental data, to tie together information from distributed sources, and to pursue other scientific endeavors. Out of necessity, these complex applications need to execute in distributed environments and make use of a number of heterogeneous resources. In this paper we describe some of these applications and illustrate techniques that improve their performance and reliably in distributed environments, such as grids and clouds. Although clouds were first introduced in the business arena, they have a potential to be provide on-demand resources for scientific computations.

Key words: scientific workflows, grid computing, cloud computing, applications, heterogeneous distributed environments

The International Journal of High Performance Computing Applications,
Volume 00, No. 0, xxxxxx 2009, pp. 000–000
DOI: 10.1177/1094342009356432

1 Introduction

Many scientific applications are "born distributed"; data are collected at different locations, stored, and replicated across wide area networks, software components are developed to target different execution environments, computing resources and services are distributed and heterogeneous, and collaborators share their expertise and pursue research goals while being separated by geographic distance. Scientific workflows are being used to bring together these various resources and answer complex research questions. They describe the relationship of the individual computational components and their input and output data in a declarative way. In astronomy, scientists are using workflows to generate science-grade mosaics of the sky (Montage, n.d.), to examine the structure of galaxies (Taylor et al. 2003), and in general to understand the structure of the universe. In bioinformatics, they are using workflows to understand the underpinnings of complex diseases (Stevens et al. 2003; Oinn et al. 2006). In earthquake science, workflows are used to predict the magnitude of earthquakes within a geographic area over a period of time (Deelman et al. 2006a). In physics, workflows are used to try to measure gravitational waves (Brown et al. 2006) and model the structure of atoms (Piccoli, 2008). In ecology, scientists explore the issues of biodiversity (Jones et al. 2005).

Today, workflow applications are running on the national and international cyberinfrastructure, such as the Open Science Grid (OSG, n.d.), the TeraGrid (2004), Enabling Grids for E-sciencE (EGEE, n.d.), and others. These infrastructures allow access to high-performance resources over wide area networks. The broad spectrum of distributed computing provides unique opportunities for large-scale, complex scientific applications. In addition to the large-scale cyberinfrastructure, applications can target a campus cluster, or utility computing platforms such as commercial (Amazon Elastic Compute Cloud, n.d.; Google App Engine, n.d.) and academic clouds (Nimbus Science Cloud, n.d.). However, these opportunities also bring with them many challenges. It is hard to decide which resources to use and for how long. It is hard to determine what the cost/benefit tradeoffs are when running in a particular environment. It is also difficult to reach good performance and reliability for an application on a given system.

In this paper, we describe a number of scientific workflow applications, their goals, and the challenges they face executing in a distributed environment. We also describe a number of techniques that have been used to achieve good performance and reliability in grid environments and reflect on their use in clouds. Finally, we examine some open questions.

The rest of the paper is organized as follows. Sections 2 and 3 describe a number of workflow-based applications, their characteristics, and the execution environment they target. Section 4 describes how the applications are mapped

Table 1
Characteristics of Montage Workflows of Different Sizes.

Size of the mosaic in degrees square	Number of input data files	Number of jobs	Number of intermediate files	Total data footprint	Approx. execution time (20 procs)
1	53	232	588	1.2 GB	40 min
2	212	1,444	3,906	5.5 GB	49 min
4	**747**	**4,856**	**13,061**	**20 GB**	**1hr 46 min**
6	1,444	8,586	22,850	38 GB	2 hr 14 min
10	3,722	20,652	54,434	97 GB	6 hr

onto the distributed environment. The challenges involved in the mapping and execution are described in Section 5. Section 6 describes related work and Section 7 concludes the paper.

2 Workflow Applications

There are a growing number of scientific workflow applications (Bharathi et al. 2008) and workflow systems (Taylor et al. 2006; Deelman et al. 2008a). In this section we describe some applications that make use of scientific workflow technologies and, in particular, of our Pegasus Workflow Management System (Pegasus-WMS) (Deelman et al. 2005, 2006b; Pegasus, n.d.).

2.1 Providing a Service to a Community

Montage (Berriman et al. 2003, 2004; Montage, n.d.) is an astronomy application that generates science-grade mosaics of the sky, based on the user request. The goal of the Montage application is to make raw data (images of the sky) and derived data (mosaics of images) available to a broad range of users. These can be professional or amateurs astronomers. Montage has its own computing resources that include cluster and a storage system to archive the raw data and requested mosaics. Thus, a number of mosaic requests can be handled locally. However, as the system becomes more successful and users request more images, or for large requests (for large areas of the sky) the Montage project needs to rely on shared cyberinfrastructure resources to provide the corresponding capabilities. Recently, the project also started investigating the possibility and costs of running the workflows on utility computing environments, such as the Amazon Cloud (Berriman et al. 2008; Deelman et al. 2008b). Although the current Amazon pricing structure is prohibitively expensive to host large (12 TB) data archives (at the cost of US$1,800 per month), it can be used to process large-scale mosaics at reasonable costs (~US$10).

The mosaics are constructed using a workflow whose characteristics are shown in Table 1. The inputs to the workflow include the input images in standard Flexible Image Transport System (FITS, n.d.) format (a file format used throughout the astronomy community) taken from image archives, such as the Two Micron All Sky Survey (2MASS) (Skrutskie et al. 1997), and a "template header file" that specifies the mosaic to be constructed. The input images are first reprojected to the coordinate space of the output mosaic, the reprojected images are then background rectified and finally coadded to create the final output mosaic (see Figure 1).

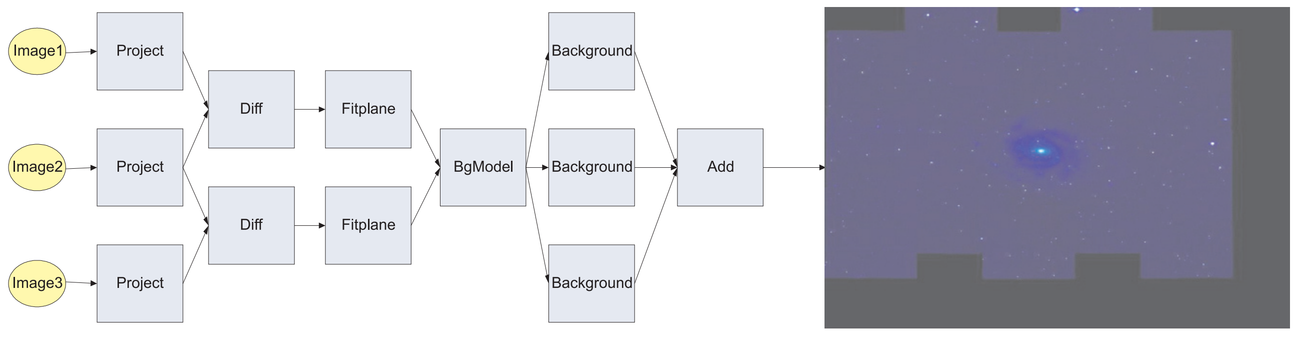


Fig. 1 A sketch of the montage workflow.

Table 2
Data and CPU Requirements for the CyberShake Components, per Site of Interest.

Component	Computation type	Data	CPU hours
Mesh generation	MPI	15 GB	150
SGT simulation	MPI	40 GB	10,000
SGT extraction	Single processor	1 GB	250
Seismogram synthesis	Single processor	10 GB	6,000
PSA calculation	Single processor	90 MB	100
Totals		66 GB	17,000

2.2 Supporting Community-based Analysis

The Southern California Earthquake Center (SCEC, 2006) project uses workflow technologies to develop physically based shake maps of the Southern California area. SCEC researchers are developing physics-based models of earthquake processes and integrating these models into a scientific framework for seismic hazard analysis and risk management. These seismic simulations are leading toward a physics-based, model-oriented approach to earthquake science with the eventual goal of transforming seismology into a predictive science with forecasting capabilities similar to those available today in the areas of climate modeling and weather forecasting.

To characterize the earthquake hazards in a region, seismologists and engineers utilize the Probabilistic Seismic Hazard Analysis (PSHA) technique (Graves, 2008). PSHA attempts to quantify the peak ground motions from all possible earthquakes that might affect a particular site (geographic location) and to establish the probabilities that the site will experience a given ground motion level over a particular time frame. PSHA information is used by city planners and building engineers and is often the basis for defining building codes in a region.

Up to now, PSHA techniques have not fully integrated the recent advances in earthquake simulation capabilities. Probabilistic seismic hazard curves have been calculated using empirically based attenuation relationships that represent relatively simple analytical models based on the regression of observed data. However, it is widely believed that significant improvements in PSHA will rely on the use of more physics-based waveform modeling.

The goal of the CyberShake project (Maechling et al. 2006) is to bring the physics-based modeling for PSHA calculation into the mainstream. CyberShake calculations consist of two main parts. Firstly, a message-passing interface (MPI)-code is run to calculate volumetric datasets called strain Green tensors (SGTs). Secondly, a large-scale post-processing calculation is done in which hundreds of thousands of serial jobs process the SGTs. The serial jobs often have short run times on the order of seconds. The SCEC project uses Pegasus to run CyberShake workflows on the National Science Foundation (NSF) TeraGrid. A single CyberShake workflow consists of approximately 800,000 jobs (Callaghan et al. 2008). Table 2 summarizes the data footprint and computational requirements of the CyberShake workflows.

The workflow characteristics of SCEC-like applications are that codes are collaboratively developed by a number of project scientists. These codes are then “strung” together to model complex systems. Some of the main challenges for these types of applications are the ability to correctly connect components, not only at a syntactic but also at a semantic level, and to be able to achieve scalability, both in the size of workflows that can be run and in the number of workflows that constitute the greater analysis.

2.3 Processing Large Amounts of Shared Data on Shared Resources

Today, science experiments collect large amounts of data, easily reaching Terabyte scales. One such project is the Laser Gravitational-Wave Observatory (LIGO; Barish and Weiss, 1999). LIGO is a network of gravitational-wave detectors, one located in Livingston, LA and two co-located in Hanford, WA. The observatories' mission is to detect and measure gravitational waves predicted by general relativity – Einstein's theory of gravity – in which gravity is described as due to the curvature of the fabric of time and space. One well-studied phenomenon that is expected to be a source of gravitational waves is the inspiral and coalescence of a pair of dense, massive astrophysical objects, such as neutron stars and black holes. Such binary inspiral signals are among the most promising sources for LIGO. Gravitational waves interact extremely weakly with matter, and the measurable effects produced in terrestrial instruments by their passage will be miniscule. In order to increase the probability of detection, a large amount of data, which contains the strain signal that measures the passage of gravitational waves, needs to be acquired and analyzed. LIGO applications often require on the order of a terabyte of data to produce meaningful results (Ramakrishnan et al. 2007; Singh et al. 2007).

Data from the LIGO detectors is analyzed by the LIGO Scientific Collaboration (LSC), which possesses many project-wide computational resources. Additional resources would allow the LSC to increase its science goals. Thus, the LSC has been reaching out toward grid deployments, such as the OSG, to extend their own capabilities. A scientifically meaningful run of the binary inspiral analysis requires a minimum of 221 GB of gravitational-wave data and approximately 70,000 computational workflow tasks (Brown et al. 2006).

Some of the challenges faced by LIGO are the capture of data generated by various instruments and their cataloging in community data registries (Chervenak et al. 2005). The amounts of data that need to be processed may exceed the computational capabilities of the LIGO Data Grid and thus workflows need to be able to run in a number of different environments. At the same time, in order to be able to manage the number of computations, issues of automation, scalability, and reliability are critical.

2.4 Automating the Work of Single Scientist

Members of smaller organizations, or even single scientists, can also benefit from workflow technologies. In particular, issues such as the automation of data processing, as well as obtaining flexibility and maintenance of the data processing pipelines, are very important. An example of such a project is the University of Southern California (USC) Epigenome Center, which is analyzing human DNA sequences (Juve et al. 2009). The Center is currently using the Illumina Genetic Analyzer (GA) system to generate high throughput DNA sequence data (up to 8 billion nucleotides per week) to map the epigenetic state of human cells on a genome-wide scale.

The Center is using workflow technologies to support these epigenomic sequencing efforts. The workflow shown in Figure 2 consists of seven basic steps that (1) transfer sequence data to the cluster storage system, (2) split sequence files into multiple parts to be processed in parallel, (3) convert sequence files to the appropriate file format, (4) filter out noisy and contaminating sequences, (5) map sequences to their genomic locations, (6) merge output from individual mapping steps into a single global map, and (7) use sequence maps to calculate the sequence density at each position in the genome. The Center is currently using this workflow to process its production of DNA methylation and histone modification data. While the workflow currently implements the minimum requirements to effectively analyze the data, additional planned quality control and checkpoint steps will make the pipeline more robust.

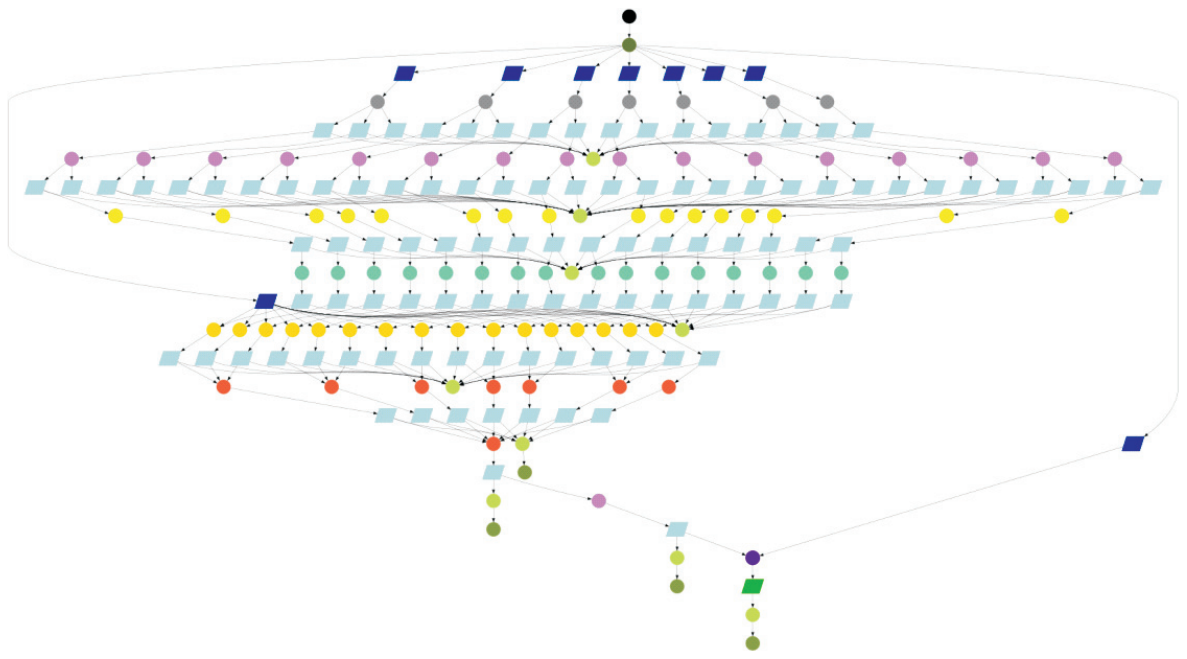


Fig. 2 Epigenomic workflow (computational jobs are shown as circles, data transfer jobs as rhomboids).

Table 3
Characteristics of Four Different Applications.

	Number of workflow tasks	Average task runtime	Size of input data	Size of intermediate data	Size of output data
CyberShake	~800,000	100s of seconds	1 GB	10s of GB	100s of MBs
Epigenomics	~600	450 tasks at 1–10 s, 150 tasks at 160 min	~400 MB	3 GB	~3 MB
LIGO's inspiral	~70,000	From 10s of seconds to 100s of seconds	~700 MB up to ~1.5 TB	~10 MB up to ~200 MB	~10 MB up to ~200 MB
Montage (1 degree square)	~200	10s of seconds	100s of MB	1 GB	350 MB

In addition to automation and flexibility in analysis modification, scientists are using workflow technologies to automatically record how the data was produced (track its provenance; IPAW, 2006).

2.5 High-level Application Characteristics

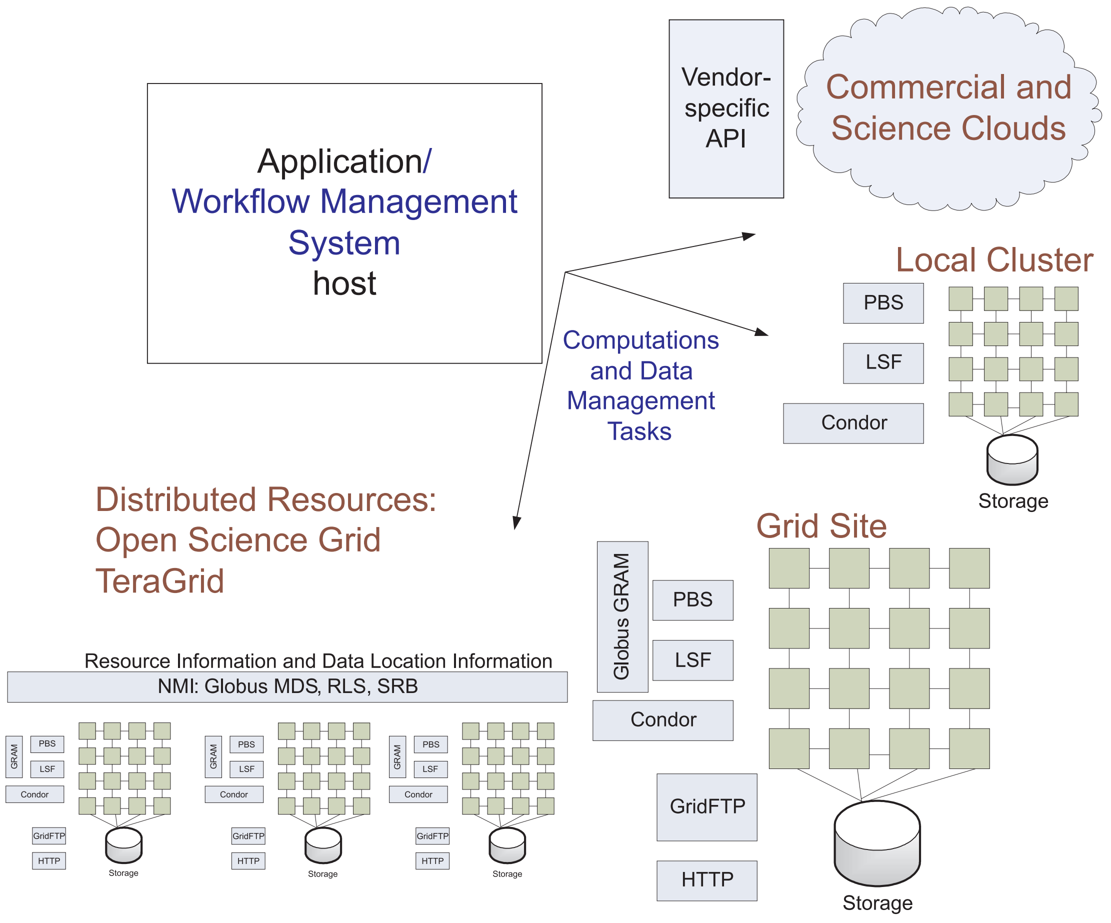
Although these applications differ greatly and their computational characteristics can differ based on the problem size, Table 3 shows task and data characteristics for some typical workflow configurations. These do not mean to represent exact numbers for individual tasks (as in Bharathi et al. (2008)), but rather are coarse-grained approximations.

Obviously, optimizing the execution of this wide range of applications can be difficult. However, some challenges are common: dealing with short-running jobs that need to be sent for execution (typically over the wide area network) to a batch resource, managing the data used and generated during the execution, and of course providing reliability. We demonstrate some approaches to these challenges in Section 5.

3 Execution Environment

Today's applications are running on local clusters, campus clusters, grids, or more recently science and commercial clouds. Applications access their resources either directly or across the wide area network, using remote submission interfaces and data transfer mechanisms (see Figure 3). In this figure we distinguish the application host that resides in the user domain and the other systems that form the shared cyberinfrastructure. The grid provides different modes of resource usage. For example in OSG, when a project joins the collaboration, it contributes some of its resources to the overall collaboration while being able to tap into the capabilities provided by other members of the community. The resource provider still has control over their own resources and may decide on how to share them with others. Providers can also potentially gain the capacity contributed by other members of the collaboration. This system works on the principle that not all resources are needed at the same time, and when a project does not need their own resources, these cycles are made available to others in the broader collaboration. Another model of computing is delivered by the TeraGrid, which is a national-level effort to provide a large-scale computational platform for science. Instead of funding individual clusters for individual science projects, it pools together the financial resources of NSF to deliver high-performance computing to a broad range of applications and communities. Research projects can apply for allocations of compute cycles that allow them to execute jobs on particular clusters or across the TeraGrid resources.

The new cloud technologies can also potentially provide benefits to today's science applications. Clouds have recently appeared as an option for on-demand computing. Originating in the business sector, clouds can provide computational and storage capacity when needed, which can result in infrastructure savings for a business. When using the cloud, applications pay only for what they use in terms of computational resources, storage, and data transfer in and out of the cloud. Clouds are also emerging in the academic arena, providing a limited number of computational platforms on demand (Cumulus (Wang et al. 2008), Eucalyptus (Nurmi et al. 2008), Nimbus (n.d.), OpenNebula (Moreno-Vozmediano et al. 2009), etc.) These Science Clouds present a great opportunity for researchers to test out their ideas and harden their codes before investing more significant resources and money into the potentially larger-scale commercial infrastructure. Through the use of virtualization, clouds can be made to look like a grid site, as they can be configured (with additional work and tools) to look like a remote cluster, presenting interfaces for remote job submission and data stage-in. As such, scientists can use their existing grid software and tools to get their science done.



Vendor-specific API
Commercial and Science Clouds
Application/ Workflow Management System host
Local Cluster
PBS
LSF
Condor
Storage
Computations and Data Management Tasks
Distributed Resources: Open Science Grid TeraGrid
Grid Site
Globus GRAM
PBS
LSF
Condor
GridFTP
HTTP
Storage
Resource Information and Data Location Information
NMI: Globus MDS, RLS, SRB
GRAM
GridFTP
HTTP

Fig. 3 Overview of the distributed execution environment.

Virtualization also 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 modeling, 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 for fear of breaking their scientific quality. Clouds and their use of virtualization technologies may make these legacy codes much easier to run. Now, the environment can be customized with a given operating system (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. However, in order to make good use of such environments, scientists need to be able to figure out how many resources they need, for how long, and what would be the associated costs.

Another interesting and differentiating aspect of the cloud is that by default it includes resource provisioning as part of the usage mode. This can potentially improve the performance of workflow-based applications. Although provisioning is sometimes made available on the grid, it is often done via verbal agreements between users and resource providers. When using automated resource-provisioning techniques, such as Condor Glide-ins (n.d.), the resource availability is bounded by the wallclock time allowed on the resource and thus can be very limiting. Thus, for example, running persistent services on the grid is difficult; this is much easier done on the cloud where there are no time limits on the use of the resources.

Table 4 shows the execution environments used by the applications described in this paper. Although these appli-

Table 4
Example Application Execution Environments.

	Local/campus cluster	Community grid	National grid
Epigenomics	Yes	No	No
LIGO	Yes	LIGO grid	OSG
Montage	Yes	No	No
SCEC	Yes	No	TeraGrid

cations are not using clouds today, they are evaluating their benefits and drawbacks.

4 Mapping the Application to the Resources

Given the multitude of different execution environments, it is important to develop applications, such as scientific workflows, in a resource-independent way and then rely on tools to map these applications onto a specific execution environment. In our work, we have been developing Pegasus-WMS to do that mapping. Pegasus-WMS (Deelman et al. 2006b) is a framework that maps complex, resource-independent scientific workflow descriptions onto distributed resources. As a result, scientists do not need to worry about the details of the underlying cyberinfrastructure or the particulars of the low-level specifications required by the cyberinfrastructure middleware (Condor (Litzkow et al. 1988), Globus (n.d.)), or the Amazon EC2 application programming interface (API) (Amazon Elastic Compute Cloud, n.d.). Pegasus dynamically discovers the available execution resources, and performs a mapping based on the execution requirements of the codes and data. Pegasus uses different scheduling algorithms to make these choices, based on information on the predicted execution time of the tasks and data access as well as on information about the resources. Pegasus can target resources such as campus clusters, the TeraGrid (2004), OSG (n.d.), and others. Pegasus relies on the Condor Directed Acyclic Graph Manager (DAGMan) workflow engine (DAGMan, 2004; Couvares et al. 2006) to launch workflow tasks and maintain the dependencies between them. Recently, Pegasus has been used to map workflows onto clouds, such as the Amazon EC2 (Juve et al. 2009; Amazon Elastic Compute Cloud, n.d.) and the Nimbus Science Cloud (Hoffa et al. 2008; Nimbus Science Cloud, n.d.). As part of the mapping, Pegasus automatically manages data generated during workflow execution by staging them out to user-specified locations, by registering them in data catalogs, and by capturing their provenance information.

Sometimes workflows as structured by scientists are not tuned for performance. In addition, given that at the time of the workflow generation the eventual execution resources are not known, it is impossible to optimize the runtime of the overall workflow. Since Pegasus dynamically discovers the available resources and their characteristics, and queries for the location of the data (potentially replicated in the environment), it improves the performance of applications through: data reuse to avoid duplicate computations and to provide reliability, workflow restructuring to improve resource allocation, and automated task and data transfer scheduling to improve overall workflow runtime. Pegasus also provides reliability through dynamic workflow remapping when failures during execution are detected. Currently, Pegasus-WMS is able to map and execute earthquake science workflows composed of a million of tasks on today's cyberinfrastructure, such as the TeraGrid (Callaghan et al. 2008).

5 Selected Challenges in Executing Workflows on Cyberinfrastructure

There are many challenges in executing workflows in distributed environments. We can think of challenges related to:

1. timely and accurate information gathering about the distributed resources;
2. efficient and scalable management of the workflow mapping and execution processes;
3. dealing with the limitations of the existing middleware or hardware (for example available memory);
4. efficient workflow tasks scheduling onto batch systems;
5. managing data transfer and storage;
6. application and resource failures.

Here we illustrate some issues and solutions when dealing with these challenges and refer to these issues throughout the remainder of this section.

5.1 Performance

The size of workflows such as CyberShake taxes the memory available on the application host. That machine is

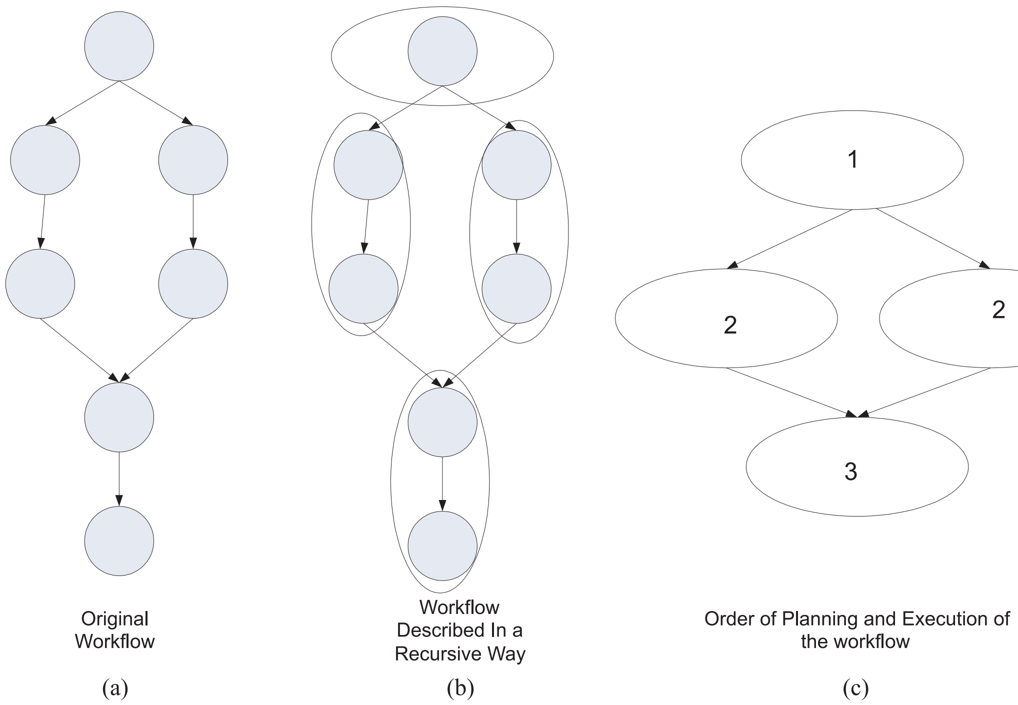

1
2
2
3
Original Workflow
Workflow Described In a Recursive Way
Order of Planning and Execution of the workflow
(a)
(b)
(c)

Fig. 4 Providing hierarchical structure to Pegasus-WMS workflows.

used to map the workflow onto the resources and to manage its execution. One particular issue arises when trying to parse the workflow specification. Because the workflow is specified in extensible markup language (XML), the in-memory parsing operation can fail or take a very long time due to limited memory (issues 2 and 3 above). In order to overcome these issues, the workflow can be partitioned (possibly recursively) so that only portions of the workflow are planned at a time. The user can thus describe their workflow in a hierarchical way and Pegasus-WMS maps and executes the resulting sub-workflows. Figure 4 illustrates this approach. In Figure 4(a), we see an original workflow without a hierarchical structure. For such a workflow the Pegasus-WMS would plan the workflow and execute all the jobs in the order indicated in the workflows. Figure 4(b) shows the same workflow but described with one level of nesting. Figure 4(c) is an illustration of how the Pegasus-WMS would map and execute the nested workflow. It would plan and execute the subworkflow #1 (which contains a single task). After its successful execution, it would then in parallel plan and execute the subworkflows labeled #2, each containing two tasks. After these tasks finished it would plan and execute the last subworkflow (#3). Since Pegasus-WMS now works on smaller portions of the workflow at a time, much larger workflows, on the order of a million of jobs, can now be handled. As a result, SCEC is able to run their workflows at the desired scale (Callaghan 2008).

The size of the workflows and the number of jobs that need to be managed on the application host can often bring up the load on that machine to unacceptable levels (issue 3). One technique that has proven beneficial in this case is to cluster workflow jobs together into larger entities. This clustering is performed by Pegasus when it maps computational workflow tasks onto the execution resources. If there is a set of tasks (destined for the same execution site) that can be clustered together without breaking workflow dependencies (if the path between any two tasks in a cluster is fully contained within the cluster), then they can be treated as one task at the application host, as one task for the purposes of sending the jobs to the remote cluster and as one task in the queue at the remote scheduler. The clustered tasks are wrapped by a script that can invoke the tasks sequentially or in parallel (according to the tasks inter-dependencies) at the remote end.

Task clustering not only reduces the load on the machine handling remote job submission, but it also has been shown to improve the overall workflow performance for fine computational granularity applications by as much as

90% (Singh et al. 2008) (issue 4). Although so far clustering had been studied on the grid, clustering techniques would also be applicable in the cloud environment. Although users would typically not compete for the same resource (so queue delays would be small), the overheads of handling many small tasks would cause performance degradation. In addition, some computing systems are configured to view the submission of a large number of jobs or data transfer requests as denial of service attacks and automatically block the user executing the workflow.

Deciding on the right size of the clusters is still a challenging problem. If clusters are too small, the benefits of clustering are limited as the number of jobs that need to be managed is not significantly reduced. If clusters are too large then the workflow can be more vulnerable to failures. If a failure occurs within a cluster, then the entire cluster needs to be re-computed.

5.2 Resource Provisioning

Another technique used to improve the overall workflow performance (issue 4) is resource provisioning (Sfiligoi, 2007; Walker and Guiang, 2007; Juve and Deelman, 2008; Walker, 2008; condor_glidein, n.d.) ahead of the workflow execution. In most grid infrastructures, task execution is performed on a best-effort basis and many applications are scheduled *ad hoc* or may use a range of scheduling methods: static user-defined resource selection, random, or heuristic-based approaches (Mandal et al. 2005). None of the current methods are adequate in today's execution environments, where the execution time and availability of resources are uncertain (issue 1). By harnessing the execution environment, however, the behavior of applications can be more predictable. One important technique in this regard is resource provisioning.

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 time 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. In our previous work, we found that resources provisioned before application execution can provide the desired execution platform and result in successful application execution that improves the overall application performance (Singh et al. 2006). Scientists are also interested in exploring the capabilities of the cloud for their work. 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. (An open question in today's clouds is how many resources can be given to any one request at any given time and how fast.)

5.3 Reliability

The importance of data management is often overlooked when managing applications in distributed environments, yet it plays a very important role in application reliability, performance, and cost (issues 5 and 6). Data-intensive workflows can easily fail, because there is not enough space to stage the needed data or to save the results. Even if a workflow executes successfully, the transfer of results back to the user can often fail. In cloud environments, it is important to estimate the amount of data needed to be provisioned in a virtual image so that the workflow has "room to grow". The cost of storage and data transfer also needs to be considered. The cost of storing data in commercial cloud storage solutions and transfers of data in and out of the cloud can become significant over time.

Sometimes, solutions to the problems can be simple. For example, for data-intensive workflows that generate large numbers of small files, zipping these files at the execution site before sending the data over the network can greatly improve the reliability of data transfers over the wide area network (Callaghan et al. 2008).

In other cases, more complex solutions need to be developed. For example, one solution to minimizing the amount of disk storage needed by a workflow is to remove the data from the execution sites as soon as it is not needed. In the extreme, data can be removed as soon as they are consumed by a workflow component (see remote input/output (I/O) in the following section). However, this may require that data be staged in again if another workflow component needs them as well. As a result, overall workflow performance may suffer. Another approach is to automatically analyze data usage within the workflow and remove the data when it is no longer needed by any subsequent workflow tasks (Ramakrishnan et al. 2007) (see dynamic cleanup in the following section). This approach can result in as much as 50% improvement in the workflow data footprint for some applications (Montage, for example). However, the success of the approach depends on the structure of the workflow. Some workflows access input data at the beginning of the workflow and also later on in the workflow execution (for example, the LIGO inspiral workflow (Brown et al. 2006)). As a result, the input data needs to be kept on the execution system for a longer period of time and the workflow data footprint is large. In some cases, with additional effort and more detailed analysis, the workflow can be restructured to reduce that footprint.

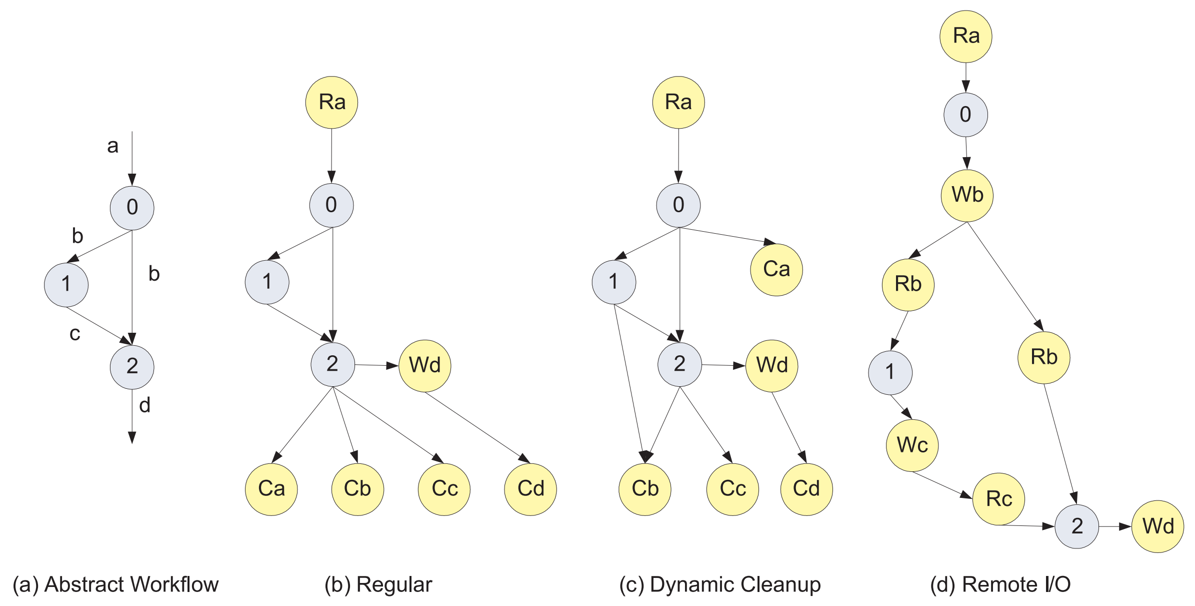


Fig. 5 Different modes of data management.

In addition to minimizing the workflow data footprint, it can be beneficial from the point of reliability to schedule workflow tasks in such a way that the task's input and output data fit into the storage available at the execution sites (issues 5 and 6). However, it is hard to get accurate and timely information about the amount of storage available to a user (storage is often shared among a collaboration and information services at the execution sites are limited). In addition, even if the amount of storage is known, its availability is not guaranteed during the execution time of a task as other users or applications can fill up the available disk space. The provisioning of storage would help with the problems; however, only limited storage reservation solutions exist in grid environments (Shoshani et al. 1998).

In commercial clouds, and in particular on the Amazon Cloud, both opportunistic and provisioned storage is available. The Amazon S3 storage service provides significant storage resources on the pay-for-what-you-use basis without storage provisioning capabilities. Amazon also provides storage provisioning capabilities via the Elastic Block Store (EBS). Storage volumes can be created in a number of sizes up to 1 TB each and mounted on the virtual computational resources. However, this storage provisioning comes at a price. Users pay for the whole amount of storage provisioned over time, independent of the amount of data stored on the volume.

5.4 Data Management and Performance Costs

In clouds, the issue of the cost of data management does not only occur in the cost of storing the data on the storage system, but also in the data transfer costs and in the CPU cost incurred by the application (issues 2 and 5). Deelman et al. (2008b) examined the cost of three different workflow data management strategies developed for Pegasus (see Figure 5). These include the following:

- **Regular**: When the compute resources used by the tasks in the workflow have access to shared storage, it can be used to store the intermediate files produced in the workflow. For example, once task 0 (Figure 5(b)) has finished execution and produced the file *b*, we allow the file *b* to remain on the storage system to be used as input later by tasks 1 and 2. In fact, the workflow manager does not delete any files used in the workflow until all the tasks in the workflow have finished execution. After that file *d*, which is the net output of the workflow, is staged out to the application/user and all files *a–d* are deleted from the storage resource. This execution mode assumes that there is shared storage that can be accessed from the compute resources used by the tasks in the workflow. This is true in the case of the Amazon system where the data stored in the S3 storage resources can be accessed from any of the EC2 compute resources.

- **Dynamic cleanup**: In the regular mode, there might be files occupying storage resources even when they have outlived their usefulness. For example, file *a* is no longer required after the completion of task 0, but it is kept around until all the tasks in the workflow have finished execution and the output data is staged out. In the dynamic cleanup mode, we automatically delete files from the storage resource when they are no longer required. This is done by Pegasus by performing an analysis of data use at the workflow level (Ramakrishnan et al. 2007). Thus, file *a* would be deleted after task 0 has completed; however, file *b* would be deleted only when task 2 has completed (Figure 5(c)). The dynamic cleanup mode potentially reduces the storage used during the workflow and thus can save money
- **Remote I/O (on-demand)**: For each task we stage the input data to the resource, execute the task, stage out the output data from the resource and then delete the input and output data from the resource (Figure 5(d)). This is the model to be used when the computational resources used by the tasks have no shared storage. For example, the tasks are running on hosts in a cluster where they have only a local file system and no network file system. This is also equivalent to the case where the tasks are doing remote I/O instead of accessing data locally.

Although the amount of disk space used by the remote I/O is the smallest of the three strategies, the amount of data transfers that occur in the workflow is costly (because of fees for transferring data to/from the cloud) and slows down the overall workflow execution. Thus, the workflow application incurs greater CPU and data transfer costs. Since CPU costs are the dominant factor in the overall execution cost of many applications, the overall cost of the workflow is the greatest for the remote I/O case. Although there are savings in the cost of storage when doing dynamic cleanup versus the regular mode of execution, these costs are relatively insignificant compared to the overall workflow cost. Thus, for many applications running on clouds, using the regular mode of data management may be best in general.

5.5 Managing Multiple Workflows

Up to now, most workflow management systems manage a single workflow at a time. However, much of scientific enquiry does not rely on the execution of a single workflow, but rather on the execution and synthesis of results from many workflows (issue 2). Thus, it is important to develop new systems that can provide the management of any number of simultaneous workflows. In our work, we developed an initial version of the Ensemble Manager (EM, n.d.) that accepts collections workflows – or workflow ensembles that define a single analysis. Simple priorities can be assigned to the individual workflows and to the workflow ensembles to give preference to the most pressing computations. For example, in the case of hurricane modeling workflows (Plale et al. 2006), some hurricane paths are more probable than others and should be explored first. Other paths based on less likely parameters may be less critical and can wait for execution. The EM is built on top of Condor (n.d.) and provides basic functionality, including submitting, executing, and killing and monitoring of a set of workflows, as well as providing hints to the system on the relative importance of the various workflows in a collection and across collections. The EM also has the concept of a workflow start where a workflow can be submitted to the system with a given start time.

Being able to manage multiple workflows concurrently may provide the benefits of better resource usage through the ability of scheduling more tasks into under-utilized resources. This in turn may also improve the overall cost of using commercial cloud resources.

6 Related Work

There are many workflow systems in existence today, each addressing some aspect of the workflow management problem (Taylor et al. 2006; Deelman et al. 2008a). Workflow systems can be broadly divided into ones that support the composition of standalone applications and those that support the composition of services. Much of the work in industry has focused on the definition of standard workflow languages, such as Business Process Execution Language (BPEL; Andrews et al. 2003), and on the development of workflow engines capable of executing BPEL-specified workflows (Active BPEL Engine, 2007; Microsoft Workflow Foundation, 2007). Some science disciplines, in particular bioinformatics, are using BPEL and BPEL-like workflow specifications to orchestrate the services that are available in their communities (Miles et al. 2004; Oinn et al. 2006; Slominski, 2006).

Expressing applications as services is not applicable in many scientific disciplines, where performance and scalability are critical and where the service invocation overheads are simply too expensive. Thus, the workflows are often expressed as workflows of standalone applications. There are many workflow systems that support application component composition and execution. Some, such as Triana (Churches et al. 2006), Kepler (Ludäscher et al. 2005), and VisTrails (Callaghan et al. 2006), provide graphical user interfaces for workflow composition and some, such as Karajan (Laszewski and Hategan, 2006), provide a scripting language to specify the workflow. Some of these workflow systems support more complex control structure loops, conditionals, and hierarchical workflow definitions. However, in our hierarchical workflow we can not only capture the workflow structure (as in the other systems), but we also use it to order and time the mapping of the portions of the workflow to the computational resources.

Today, not many workflow systems perform workflow-level optimization and resources scheduling; rather they rely on the user to select resources or services. Among such workflow systems are Kepler, Taverna, and VisTrails. However, in the case of Taverna, the user can provide a set of services that match a particular workflow component, so if errors occur, an alternate service can be automatically invoked. Some workflow management systems, such as P-GRADE (Kertész et al. 2006), use an external broker to schedule tasks onto resources. The Askalon system (Wieczorek et al. 2005), designed to support task-level workflows, has a rich environment for mapping workflows onto resources. It not only does the resource assignment, but can also automatically provision the resources ahead of the workflow execution. Askalon contains two major components responsible for workflow scheduling: the scheduler and the resource management system (GridARM). GridARM serves as a data repository that provides the scheduler with all the information needed for scheduling, including the available resources and the applications deployed on the Grid. Apart from the basic functionalities, GridARM can also provide more advanced resource and application discovery techniques based on quality-of-service matching, and it can guarantee advance reservation of resources. In order to support the scheduler, Askalon has developed a performance analysis and prediction system that can estimate the runtime of the workflow tasks, as well as the data transfer times of data between tasks. The scheduler makes full-graph scheduling of scientific workflows. Askalon can also support workflow transformation techniques to improve performance. However, unlike our approach, Askalon focuses on simplifying the conditional structures present in its workflows. Among the optimization transformations are techniques employed in parallel compilers and include branch prediction, parallel loop unrolling, and sequential loop elimination. If the choices made during the transformation (such as branch prediction) are erroneous, the workflow is adjusted and rescheduled at runtime. Once the Directed Acyclic Graph (DAG) is created, it is mapped onto the available resources based on a scheduling algorithm.

7 Conclusions

In this paper we delineated some of the challenges faced by workflow applications when they are executing in distributed environments, such as the grid or more recently the cloud. We based our observations on our experiences with a number of scientific applications and with the Pegasus-WMS. Although much progress has been made to make the execution of applications more efficient and more reliable, major challenges still remain. High-level systems, such as workflow systems, can mask some of the complexity of the underlying infrastructure through abstract interfaces and fault-recovery mechanism. However, the latter are still limited and face the difficulties of interpreting the faults occurring throughout the distributed system and through the many levels of software used to manage them. Thus, at times, the user is faced with a number of cryptic messages being sent from distributed middleware and is often unable to determine what went wrong. Thus, it is critical that more work be done in the area of improving middleware and providing sophisticated, yet easy to use, debugging facilities for distributed applications. Usability also remains a big concern. Up to now, most of the users of distributed systems have been astronomers, climate modelers, physicists, and other physical scientists familiar with computing paradigms and computing systems. However, more and more researchers from less computationally focused domains are reaching out to distributed systems for their work. This move is underway because these researchers rely on distributed data and distributed collaborations to do their work. Thus, providing user-friendly and user-centered computational capabilities is becoming increasingly critical.

Acknowledgements

This work was funded by the NSF under Cooperative Agreement OCI-0438712 and grant # CCF-0725332. This work would not be possible without the collaboration with the following people: Ben Berman (USC), Bruce Berriman (Caltech), Kent Blackburn (Caltech), Duncan Brown (Syracuse University), Britta Daudert (Caltech), Gideon Juve (USC), Phil Maechling (USC), Gaurang Mehta (USC), Karan Vahi (USC), and many others.

Author's Biography

Ewa Deelman is a Research Associate Professor at the USC Computer Science Department and a Project Leader at the USC Information Sciences Institute (ISI). Her research interests include the design and exploration of collaborative, distributed scientific environments, with particular emphasis on workflow management as well as the management of large amounts of data and metadata.

At the ISI, she is leading the Pegasus project, which designs and implements workflow mapping techniques for large-scale workflows running in distributed environments. She received her PhD from Rensselaer Polytechnic Institute in Computer Science in 1997 in the area of parallel discrete event simulation.

References

Active BPEL Engine. (2009). http://www.activevos.com/community-open-source.php

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

Andrews, T., Curbera, F., Dholakia, H., Goland, Y., Klein, J., Leymann, F., Liu, K., Roller, D., Smith, D., Thatte, S., Trickovic, I. and Weerawarana, S. (2003). Specification: business process execution language for Web Services Version 1.1, http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/

Barish, B. C. and Weiss, R. (1999). LIGO and the detection of gravitational waves. *Phys Today* **52**: 44.

Berriman, B., Bergou, A., Deelman, E., Good, J., Jacob, J., Katz, D., Kesselman, C., Laity, A., Singh, G., Su, M.-H. and Williams, R. (2003). Montage: a grid-enabled image mosaic service for the NVO. In the Proceedings of Astronomical Data Analysis Software & Systems (ADASS) XIII.

Berriman, G. B., Deelman, E., Good, J., Jacob, J., Katz, D. S., Kesselman, C., Laity, A., Prince, T. A., Singh, G. and Su, M.-H. (2004). Montage: a grid enabled engine for delivering custom science-grade mosaics on demand. In Proceedings of SPIE Conference 5487: Astronomical Telescopes.

Berriman, G. B., Good, J. C., Deelman, E., Singh, G. and Livny, M. (2008). A cost benefit study of doing astrophysics on the cloud: production of image mosaics. In the Proceedings of Astronomical Data Analysis Software and Systems (ADASS).

Bharathi, S., Chervenak, A., Deelman, E., Mehta, G., Su, M.-H. and Vahi, K. (2008). Characterization of scientific workflows. In Proceedings of the 3rd Workshop on Workflows in Support of Large-Scale Science (WORKS08).

Brown, D. A., Brady, P. R., Dietz, A., Cao, J., Johnson, B. and McNabb, J. (2006). A case study on the use of workflow technologies for scientific analysis: gravitational wave data analysis. In Workflows for e-Science, edited by I. Taylor, E. Deelman, D. Gannon and M. Shields, Springer.

Callaghan, S., Maechling, P., Deelman, E., Vahi, K., Mehta, G., Juve, G., Milner, K., Graves, R., Field, E., Okaya, D., Gunter, D., Beattie, K. and Jordan, T. (2008). Reducing time-to-solution using distributed high-throughput mega-workflows – experiences from SCEC CyberShake. In Proceedings of the 4th IEEE International Conference on e-Science (e-Science 2008), Indianapolis, IN.

Callahan, S. P., Freire, J., Santos, E., Scheidegger, C. E., Silva, C. T. and Vo, H. T. (2006). VisTrails: visualization meets data management. In the Proceedings of ACM SIGMOD.

Chervenak, A., Schuler, R., Kesselman, C., Koranda, S. and Moe, B. (2005). Wide area data replication for scientific collaborations. In Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing, pp. 1–8.

Churches, D., Gombas, G., Harrison, A., Maassen, J., Robinson, C., Shields, M., Taylor, I. and Wang, I. (2006). Programming scientific and distributed workflow with Triana services. *Concurrency Comput. Pract. Ex.* **18**: 1021–1037.

Condor. http://www.cs.wisc.edu/condor

condor_glidein. http://www.cs.wisc.edu/condor/glidein.

Couvares, P., Kosar, T., Roy, A., Weber, J. and Wenger, K. (2006). Workflow management in Condor. In Workflows in e-Science, edited by I. Taylor, E. Deelman, D. Gannon and M. Shields, Springer, pp. 357–375.

DAGMan (Directed Acyclic Graph Manager). (2004). http://www.cs.wisc.edu/condor/dagman/

Deelman, E., Callaghan, S., Field, E., Francoeur, H., Graves, R., Gupta, N., Gupta, V., Jordan, T. H., Kesselman, C., Maechling, P., Mehringer, J., Mehta, G., Okaya, D., Vahi, K. and Zhao, L. (2006a). Managing large-scale workflow execution from resource provisioning to provenance tracking: the CyberShake example. In E-SCIENCE '06: Proceedings of the 2nd IEEE International Conference on e-Science and Grid Computing, p. 14.

Deelman, E., Gannon, D., Shields, M. and Taylor, I. (2008a). Workflows and e-Science: an overview of workflow system features and capabilities. *Future Generat. Comput Syst,* doi:10.1016/j.future.2008.06.012.

Deelman, E., Mehta, G., Singh, G., Su, M.-H. and Vahi, K. (2006b) Pegasus: mapping large-scale workflows to distributed resources. In Workflows in e-Science, edited by I. Taylor, E. Deelman, D. Gannon and M. Shields, Springer.

Deelman, E., Singh, G., Livny, M., Berriman, B. and Good, J. (2008b). The cost of doing science on the cloud: the Montage example. In the Proceedings of SC'08, Austin, TX.

Deelman, E., Singh, G., Su, M.-H., Blythe, J., Gil, Y., Kesselman, C., Mehta, G., Vahi, K., Berriman, G. B., Good, J., Laity, A., Jacob, J. C. and Katz, D. S. (2005). Pegasus: a framework for mapping complex scientific workflows onto distributed systems. *Sci. Program.* **13**: 219–237.

Enabling Grids for E-sciencE (EGEE). http://www.eu-egee.org/

Ensemble Manager. http://pegasus.isi.edu/ensemble/

Flexible Image Transport System. http://fits.gsfc.nasa.gov/

Globus. http://www.globus.org

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

Graves, R. (2008). Physics based probabilistic seismic hazard calculations for Southern California. In Proceedings of the 14th World Conference on Earthquake Engineering Beijing, China.

Hoffa, C., Mehta, G., Freeman, T., Deelman, E., Keahey, K., Berriman, B. and Good, J. (2008). On the Use of cloud computing for scientific workflows. In Proceedings of the 3rd International Workshop on Scientific Workflows and Business Workflow Standards in e-Science (SWBES), in conjunction with the 4th IEEE International Conference on e-Science (e-Science 2008), Indianapolis, IN.

IPAW (2006). Provenance and Annotation of Data, International Provenance and Annotation Workshop (IPAW 2006), Chicago, IL, May 3–5, Revised Selected Papers.

Juve, G. and Deelman, E. (2008). Resource provisioning options for large-scale scientific workflows. In Proceedings of the 3rd International Workshop on Scientific Workflows and Business Workflow Standards in e-Science (SWBES), in conjunction with the 4th IEEE International Conference on e-Science (e-Science 2008), Indianapolis, IN.

Juve, G., Deelman, E., Vahi, K., Mehta, G., Berriman, B., Berman, B. P. and Maechling, P. (2009). Scientific workflow applications on Amazon EC2. In Cloud Computing Workshop in Conjunction with e-Science, Oxford, UK: IEEE.

Kertész, A., Sipos, G. and Kacsuk, P. (2006). Brokering multigrid workflows in the P-GRADE portal. In the Proceedings of Euro-Par 2006: Parallel Processing, vol. 4375, Berlin: Springer.

Laszewski, G. v. and Hategan, M. (2006). Workflow concepts of the Java CoG kit. *J. Grid Comput* **3**: 239–258.

Litzkow, M., Livny, M. and Mutka, M. (1988). Condor – a hunter of idle workstations. In Proceedings of the 8th International Conference on Distributed Computing Systems, pp. 104–111.

Ludäscher, B., Altintas, I., Berkley, C., Higgins, D., Jaeger-Frank, E., Jones, M., Lee, E., Tao, J. and Zhao, Y. (2005). Scientific workflow management and the Kepler system. *Concurrency Comput. Pract.* **18**: 1039–1065.

Maechling, P., Deelman, E., Zhao, L., Graves, R., Mehta, G., Gupta, N., Mehringer, J., Kesselman, C., Callaghan, S., Okaya, D., Francoeur, H., Gupta, V., Cui, Y., Vahi, K., Jordan, T. and Field, E. (2006). SCEC CyberShake workflows—automating probabilistic seismic hazard analysis calculations. In Workflows for e-Science, edited by I. Taylor, E. Deelman, D. Gannon and M. Shields, Springer.

Mandal, A., Kennedy, K., Koelbel, C., Marin, G., Mellor-Crummey, J., Liu, B. and Johnsson, L. (2005). Scheduling strategies for mapping application workflows onto the grid. In Proceedings of the IEEE International Symposium on High Performance Distributed Computing (HPDC).

Microsoft Workflow Foundation. (2007). http://msdn2.microsoft.com/en-us/netframework/aa663328.aspx

Michener, W., Beach, J., Bowers, S., Downey, L., Jones, M., Ludäscher, B., Pennington, D., Rajasekar, A., Romanello, S., Schildhauer, M., Vieglais, D. and Zhang, J. (2005). Data Integration and Workflow Solutions for Ecology. Data Integration in the Life Sciences (DILS), San Diego, California, USA, July 20–22, 2005, pp. 321–324: Springer.

Miles, S., Papay, J., Wroe, C., Lord, P., Goble, C. and Moreau, L. (2004). Semantic description, publication and discovery of workflows in myGrid. Technical Report ECSTR-IAM04-001, Electronics and Computer Science, University of Southampton.

Moreno-Vozmediano, R., Montero, R. and Llorente, I. (2009). Elastic management of cluster-based services in the cloud. In Proceedings of the 1st Workshop on Automated Control for Datacenters and Clouds (ACDC09), pp. 19–24.

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

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

Nurmi, D., Wolski, R., Grzegorczyk, C., Obertelli, G., Soman, S., Youseff, L. and Zagorodnov, D. (2008). The Eucalyptus Open-source Cloud-computing System. In the Proceedings of Cloud Computing and its Applications, Chicago.

Oinn, T., Li, P., Kell, D. B., Goble, C., Goderis, A., Greenwood, M., Hull, D., Stevens, R., Turi, D. and Zhao, J. (2006). Taverna/myGrid: aligning a workflow system with the life sciences community. In Workflows in e-Science, edited by I. Taylor, E. Deelman, D. Gannon and M. Shields. Springer.

Open Science Grid. www.opensciencegrid.org

Pegasus. http://pegasus.isi.edu

Piccoli, L. (2008). Lattice QCD workflows: a case study. In SWBES08: Challenging Issues in Workflow Applications, Indianapolis, IN.

Plale, B., Gannon, D., Brotzge, J., Droegemeier, K., Kurose, J., McLaughlin, D., Wilhelmson, R., Graves, S., Ramamurthy, M. and Clark, R. (2006). CASA and LEAD: adaptive cyberinfrastructure for real-time multiscale weather forecasting. *Computer* **39**: 56–64.

Ramakrishnan, A., Singh, G., Zhao, H., Deelman, E., Sakellariou, R., Vahi, K., Blackburn, K., Meyers, D. and Samidi, M. (2007). Scheduling data-intensive workflows onto storage-constrained distributed resources. In Proceedings of the 7th IEEE International Symposium on Cluster Computing and the Grid — CCGrid 2007.

Sfiligoi, I. (2007). glideinWMS—a generic pilot-based workload management system. Fermi Lab. doi:10.1088/1742-6596/119/6/062044.

Shoshani, A., Bernardo, L. M., Nordberg, H., Rotem, D. and Sim, A. (1998). Storage management for high energy physics applications. In the Proceedings of Computing in High Energy Physics 1998 (CHEP 98).

Singh, G., Kesselman, C. and Deelman, E. (2006). Application-level resource provisioning on the grid. In the Proceedings of e-Science, Amsterdam, The Netherlands.

Singh, G., Su, M. H., Vahi, K., Deelman, E., Berriman, B., Good, J., Katz, D. S. and Mehta, G. (2008). Workflow task clustering for best effort systems with Pegasus. Proceedings of the 15th ACM Mardi Gras conference: From lightweight mash-ups to lambda grids: Understanding the spectrum of distributed computing requirements, applications, tools, infrastructures, interoperability, and the incremental adoption of key capabilities.

Singh, G., Vahi, K., Ramakrishnan, A., Mehta, G., Deelman, E., Zhao, H., Sakellariou, R., Blackburn, K., Brown, D., Fairhurst, S., Meyers, D., Berriman, G. B., Good, J. and Katz, D. S. (2007). Optimizing workflow data footprint. *Sci. Program.* **15**: 249–268.

Skrutskie, M. F., Schneider, S. E., Stiening, R., Strom, S. E., Weinberg, M. D., Beichman, C., Chester, T., Cutri, R., Lonsdale, C. and Elias, J. (1997). The Two Micron All Sky Survey (2MASS): overview and status. In The Impact of Large Scale Near-IR Sky Surveys, (Garzon F, Epchtein, N, Omont, A, Burton, B, Persi, P eds), Dordrecht: Kluwer Academic Publishers, pp. 25.

Slominski, A. (2006). Adapting BPEL to scientific workflows. In Workflows for e-Science, edited by I. Taylor, E. Deelman, D. Gannon and M. Shields, Springer.

Southern California Earthquake Center (SCEC) (2006). http://www.scec.org/

Stevens, R. D., Robinson, A. J. and Goble, C. A. (2003). myGrid: personalised bioinformatics on the information grid. In Bioinformatics (11th International Conference on Intelligent Systems for Molecular Biology), vol. 19.

Taylor, I., Deelman, E., Gannon, D. and Shields, M. eds. (2006). Workflows in e-Science, Springer.

Taylor, I., Shields, M., Wang, I. and Philp, R. (2003). Distributed P2P computing within Triana: a galaxy visualization test case. In the Proceedings of IPDPS 2003.

TeraGrid. (2004). http://www.teragrid.org/

Walker, E. (2008). Continuous adaptation for high performance throughput computing across distributed clusters. In Proceedings of the 2008 IEEE International Conference on Cluster Computing, pp. 369–375.

Walker, E. and Guiang, C. (2007). Challenges in executing large parameter sweep studies across widely distributed computing environments. In Proceedings of the 5th IEEE workshop on challenges of large applications in distributed environment, pp. 11–18.

Wang, L., Tao, J., Kunze, M., Rattu, D. and Castellanos, A. C. (2008). The Cumulus Project: build a scientific cloud for a data center. In the Proceedings of Cloud Computing and its Applications, Chicago, IL.

Wieczorek, M., Prodan, R. and Fahringer, T. (2005). Scheduling of scientific workflows in the ASKALON grid environment. *A CM SIGMOD SIGMOD Rec.* **34**.