Resource Provisioning Options for Large-Scale Scientific Workflows

Gideon Juve, Ewa Deelman
Information Sciences Institute, University of Southern California
juve@usc.edu, deelman@isi.edu

Abstract

Scientists in many fields are developing large-scale workflows containing millions of tasks and requiring thousands of hours of aggregate computation time. Acquiring the computational resources to execute these workflows poses many challenges for application developers. Although the grid provides ready access to large pools of computational resources, the traditional approach to accessing these resources suffers from many overheads that lead to poor performance. In this paper we examine several techniques based on resource provisioning that may be used to reduce these overheads. These techniques include: advance reservations, multi-level scheduling, and infrastructure as a service (IaaS). We explain the advantages and disadvantages of these techniques in terms of cost, performance and usability.

1. Introduction

Scientists in fields such as high-energy physics [6], earthquake science [3], and astronomy [25], are developing large-scale workflows. These workflows contain millions of short-duration, serial tasks and require thousands of hours of total CPU time. Acquiring the computational resources needed to execute these workflows in a reasonable amount of time is one of the major challenges facing workflow developers.

Resources for large-scale workflows are typically acquired through a Grid [11]. Grids are composed of independent sites connected via high-speed networks that provide computational resources in the form of supercomputers or space-shared clusters. Users access grid resources by submitting batch jobs via grid protocols to a local resource manager (LRM) at each site. These jobs are provided with best-effort quality of service using a queue-based provisioning model.

This approach to resource access often results in poor performance for workflow applications. The queuing and scheduling delays for grid sites can dramatically increase job runtimes, and many sites implement scheduling policies that are not favorable to workflows containing many short, serial tasks.

In order to avoid or minimize job delays, several resource provisioning options are available to workflow applications. Advance reservations enable users to allocate resources for their exclusive use for a given period of time. This technique reduces queuing delays by eliminating competition for shared resources. Multi-level scheduling [28] is a provisioning technique that enables user-level resource managers to control jobs and resources. This approach reduces queuing delays by reserving resources, and reduces scheduling delays by allowing scheduling policies to be managed at the application level. Infrastructure as a Service (IaaS) is a provisioning technique that has developed to support virtual machine grid computing [8] and cloud computing [29]. It allows users to instantiate complete, virtualized execution environments on remote clusters. This capability can be combined with multi-level scheduling to create flexible and efficient platforms for workflow execution.

The rest of this paper is organized as follows. The next section explains the traditional approach to accessing computational resources on the grid and describes the effects this approach has on the performance of workflow applications. Section 3 describes alternative resource provisioning techniques and how they can be used to improve workflow performance. Section 4 describes several challenges for these resource provisioning methods and some solutions that can be used to overcome them. Finally, section 5 summarizes our discussion.

2. Traditional resource access

The traditional method of accessing resources on the grid is to submit application jobs directly to a site's local resource manager (LRM). Although this method works well for many different types of applications, it is often inefficient for large-scale workflow applications that contain a large number of short-duration, serial jobs. A single workflow for the CyberShake application [7], for example, contains more than 840,000 jobs with an average runtime of less

than 30 seconds each [3]. Similarly, a typical workflow for the Montage application contains more than 4000 jobs with durations of less than a minute each [32].

Submitting such jobs directly to a site's LRM is not ideal for several reasons. First, because LRMs support many advanced features, such as parallel jobs, multiple queues, reservations, and job prioritization, they often have large scheduling overheads and long scheduling intervals. For particularly short jobs the scheduling delay is often longer than the runtime of the job itself. In addition, if the scheduling interval is longer than actual runtime of the job, then the effective runtime of the job experienced by the application will be increased. In workflows with many short-running jobs these overheads result in low throughput and long workflow runtimes.

Second, on shared clusters competition with other users for access to resources can produce long queue times. This is problematic for workflow applications because many workflow jobs must wait until the jobs they depend on have completed before they can be submitted to the queue. This causes queue times to be added at every level in the workflow and can significantly increase overall workflow runtime.

Finally, grid sites often have scheduling policies that are unfriendly to workflow jobs. Many sites prioritize large parallel jobs over smaller serial jobs, leading to increases in queuing delays. Some sites also place an upper limit on number of queued jobs per user, which limits the number of workflow jobs that can be executed concurrently.

3. Resource provisioning

The traditional approach to resource access in grid environments is based on a queuing model that provides best-effort quality of service. In this model jobs are queued until they can be matched with appropriate resources for execution. This approach ensures that access to resources is shared equally and fairly among all users of the system, but can result in long delays when competition between users forces jobs to wait for resources to become available. For applications with only one job, or with a few jobs that can be submitted in parallel, these delays are encountered only once. For workflow applications with complex job hierarchies and interdependencies the delays are encountered many times.

One way to improve quality of service for workflow applications is to use a model for resource allocation based on provisioning. With a provisioning model, resources are allocated for the exclusive use of a single user for a given period of time. This minimizes queuing delays because the user's jobs no longer compete with other jobs for access to resources. Furthermore, in contrast to the queuing model where resource allocation and scheduling occur on a per-job basis, the provisioning model allows resources to be allocated once and used for multiple jobs.

Provisioning is slightly more complex than queuing in that it requires users to make more sophisticated resource allocation decisions. There are two policies that can be used to guide these decisions. In *static provisioning* the application allocates all resources required for the computation before any jobs are submitted, and releases the resources after all the jobs have finished. This method assumes that the number of resources required is known or can be predicted in advance. In *dynamic provisioning* resources are allocated by the system at runtime. This allows the pool of available resources to grow and shrink according to the changing needs of the application. Dynamic provisioning does not require advanced knowledge of resource needs, but it does require policies for acquiring and releasing resources. It also relies on the ability of the provisioning system to acquire resources on-demand when they are needed, which may not be possible if the resources are shared with other users.

3.1 Advance Reservation

Advance reservation is a resource provisioning mechanism supported by many batch schedulers [23,27,14]. Users create advance reservations by requesting slots from the batch scheduler that specify the number of resources to reserve and the start and end times of the reservation. During the reservation period the scheduler only runs jobs that belong to the user on the reserved resources.

Although batch schedulers used by many resource providers have advance reservation features, few providers support the use of reservations. Singh, et al. conducted a survey of advance reservation capabilities at several grid sites [33]. They discovered that 50% of the sites surveyed did not support reservations at all, and that most of the sites that did support reservations required administrator assistance to create them. Only a few sites allowed users to create their own reservations. This lack of support makes using advance reservations time-consuming and cumbersome.

Scheduler-based advance reservations also increase resource usage costs. In many grid environments these costs are measured in service units. Users of advance reservations are typically charged a premium for dedicated access to resources. These premiums can be 20% to 100% above normal costs [30]. Furthermore, users are often forced to pay for the entire reservation, even if they are not able to use it all (e.g. if there is a failure that causes the application to abort, or if the actual runtime of the application is shorter than predicted).

An alternative to scheduler-based advance reservations is the use of probabilistic advance reservations [26]. In this method reservations are made based on statistical estimates of queue times. The estimates allow jobs to be submitted with a high probability of starting some time before the desired reservation begins. This allows "virtual reservations" to be created by adjusting the runtime of the job to cover both the time between the submission of the job and the desired reservation start time, and the duration of the reservation itself.

Unlike scheduler-based reservations, probabilistic reservations do not require special support from resource providers. However, probabilistic reservations are not guaranteed because the actual queue delay may exceed the predicted delay, and the final cost of a probabilistic reservation is difficult to predict because the actual runtime of the reservation job may exceed the desired reservation time.

3.2 Multi-level scheduling

Many of the performance issues encountered by workflow applications on the grid arise because resource providers manage both the allocation of resources and the scheduling and management of application jobs. In multi-level scheduling [28] these functions are separated, allowing a greater flexibility and more efficient resource management. Resource providers retain authority over the leasing of resources to users, but users are given control over application scheduling and job management. This division is accomplished by creating "personal clusters," which are temporary pools of computational resources acquired from the resource provider but managed by the user. Resources are acquired by submitting provisioning jobs to a grid site using standard mechanisms. Instead of running an application program, the provisioning jobs install and run guest node managers on the site's worker nodes. These node managers are configured to contact an external resource manager controlled by the user. The user is then able to submit jobs to an application-specific scheduler for execution on the provisioned resources.

Multi-level scheduling lessens many of the problems that plague workflow applications on the grid. Queuing delays are minimized because resources are provisioned and not shared, and scheduling overheads are reduced because jobs are not submitted to the site's LRM. In addition, multi-level scheduling makes workflow applications more attractive to remote schedulers by allowing users to make larger resource provisioning requests, enabling them to compete favorably with large parallel applications for resources.

Multi-level scheduling also enables users to control scheduling at the application level. Application-specific schedulers can be expressly configured to minimize overheads, and scheduling policies can be fine-tuned to fit the specific characteristics of workflow applications. Singh, et al. have demonstrated some of the benefits of using application-specific scheduling parameters for workflow execution in [32]. In addition, control over scheduling allows users to take advantage of one of the many task-scheduling heuristics that have been developed [35].

Unlike advance reservations, multi-level scheduling does not impose additional usage costs on the user. From the perspective of the resource provider, provisioning jobs look like normal application jobs and are charged at the normal rate. In addition, users only pay for resources that they use because provisioning jobs can be easily cancelled to stop usage charges.

3.3 Existing multi-level scheduling systems

Multi-level scheduling is a complex process requiring a significant amount of setup, configuration and debugging. This complexity is being addressed through the development of middleware systems that automate much of the work of provisioning resources and installing and configuring resource managers on remote grid sites. Many of these systems are built on top of existing resource managers, such as PBS [27], Sun Grid Engine [14], and Condor [22]. Leveraging off-the-shelf resource managers allows multi-level scheduling systems to benefit from the fault tolerance, policy management, security, and scalability features supported by these systems. Several systems also support advanced resource provisioning options, such as dynamic provisioning, automatic resubmission of provisioning requests, support for multiple resource providers, and load balancing between resource providers.

Many of the existing multi-level scheduling systems are based on Condor glideins [12]. Using this technique, node managers called "glideins" are created by starting Condor worker daemons on remote cluster nodes. Upon startup, the workers join a Condor pool administered by the user where they can be used to execute jobs submitted to the pool's queue. Glideins have been used successfully to reduce runtimes for several large-scale workflow applications [32,33].

Table 1 shows a comparison of the existing multi-level scheduling systems.

3.4 Infrastructure as a Service (IaaS)

IaaS systems, such as Amazon's Elastic Compute Cloud (EC2) [1], SODA [18], Virtual Workspaces [19], Cluster-on-demand [4] and Shirako [16], enable users to configure and launch virtual machines on remote hardware. These virtual machines can be

System	Resource Managers	Interfaces	Provisioning Policies	Firewall Negotiation	Resource Provider Interfaces
condor_glidein [5]	Condor	command-line	static	none	Globus
glideinWMS [31]	Condor	none (automatic)	dynamic	GCB	Globus
Glidein Service [15]	Condor	command-line, API	static	GCB	Globus
MyCluster [36]	Condor, SGE, PBS	command-line	static	manager on head node, virtual networking	Globus, PBS, SGE, LoadLeveler, LSF, Condor, EC2
Falkon [28]	custom	API	static, dynamic	manager on head node	Globus
VGES [21]	PBS	API	static	manager on head node	Globus, EC2

Table 1: Comparison of existing multi-level scheduling systems

allocated with specific amounts of CPU, memory, and disk space, and can be configured with custom software stacks that include specific operating systems, shared libraries, middleware services, and application software. IaaS and other computational service models such as Platform as a Service (PaaS) and Software as a Service (SaaS), can be broadly defined as "cloud computing."

In general, there are many advantages to using IaaS for grid computing [8]. Some of these are particularly relevant to workflow applications.

Workflow applications often require very complex execution environments that include specific operating systems, libraries, file system structures, and numerous application programs and configuration files. These environments are difficult to create on grid resources because users cannot modify many of the components their applications depend on, such as operating systems and system libraries. In addition, each grid site has a different configuration, which results in extra effort each time an application needs to be ported to a new site. Virtual machines allow the application developer to create a fully customized, portable execution environment configured specifically for their application.

OS-level virtualization also provides users with root access to the operating system. This is helpful because it allows users to configure operating system settings, install useful kernel modules, create and manage user accounts, and perform privileged operations such as mounting file systems. Control over file system management is particularly useful for workflow applications because it allows network file systems that contain application data and binaries to be mounted locally. This can be used to eliminate the expensive and cumbersome data staging operations that are required in grid environments.

IaaS can also be combined with multi-level scheduling to create efficient workflow execution platforms. Several authors have reported large startup delays for virtual machines [20,10,4,34,8]. These delays range from ten seconds to several minutes and have the same effect on workflow performance as scheduling and queuing delays in grid environments. Fortunately, the same multi-level scheduling techniques that can be used to eliminate delays on the grid can be used in the cloud as well. Multiple virtual machines can be created and grouped together into a single "virtual cluster" [4,10]. When configured with resource managers these virtual clusters are functionally identical to personal clusters that are created in grid environments.

Currently there are only a few resource providers that feature full OS-level virtualization services. These include Amazon and FlexiScale [9] in the commercial sector, and some experimental science clouds in the academic sector [29]. Given the recent surge of interest in this area, however, the shortage of providers is likely to be temporary.

Unlike grid resources, which are essentially free from the perspective of users, resources from commercial providers cost real dollars to use. Although usage charges for individual resources are small they can quickly add up when multiple resources are used to create virtual clusters. This also makes cloud resources more readily available than grid resources because commercial providers have an incentive to carry extra capacity and price-conscious users are less likely to oversubscribe resources.

Current IaaS systems can be complex to setup and use. Many systems require users to construct disk images containing bootable operating systems. Building these images is not trivial and may require significant debugging and trial and error. Once an image is created, however, it can be reused many times.

Finally, the virtualization technologies used by IaaS services impose runtime overheads on OS processes. This causes applications to suffer a small performance penalty when run on virtual machines.

This penalty has been measured to be less than 10% [8,10].

4. Challenges

4.1 Resource estimation

In order to provision resources for a workflow application, the resource requirements of the application must be formally specified. In addition to the individual characteristics of the resources, such as CPU architecture, memory and disk space, the specification must include 1) the number of resources required, 2) when the resources will be needed, and 3) how long the resources will be used. Developing such a specification involves a cost/performance tradeoff for the application developer. Acquiring extra resources can improve performance by providing more opportunities for concurrent execution, but this results in higher allocation costs and has the potential to leave some resources underutilized. Singh et al. explore this tradeoff in more detail in [33].

In current practice, resource specifications are often determined using ad-hoc methods based on workflow parallelism, empirical testing, resource availability, or educated guesses. These haphazard methods often produce specifications that result in poor application performance and low resource utilization. A better solution is to algorithmically generate specifications from workflow descriptions. The development of algorithms and heuristics to produce high-quality resource specifications is the subject of recent research by Huang, et al. [17] and Byun, et al. [2].

4.2 Networking

Multi-level scheduling systems function best when worker nodes have public IP addresses and are free to communicate with hosts outside their network. However, many resource providers conserve IP addresses by using private networks and isolate their worker nodes behind firewalls to protect against security threats. This prevents application-specific schedulers outside the resource provider's network from communicating directly with worker nodes.

One solution to this problem is to use Generic Connection Brokering (GCB) [13]. With GCB a server is started in a location that is accessible to both the worker nodes and the external scheduler to act as a connection broker. This broker is used to facilitate communications in a way that avoids making direct connections into the private network.

Another solution is to install and run the application-specific scheduler on the head node of the remote cluster [24]. This allows the scheduler to communicate directly with both the user's submit host and the cluster's worker nodes, but may violate site policies that restrict long-running, CPU-intensive processes on the head node. This technique is supported by MyCluster and the VGES personal cluster system.

5. Summary

In this paper, we discussed some of the resource provisioning challenges that affect workflow applications. Although the grid provides easy access to large pools of computational resources it is difficult for workflow applications to use these resources efficiently. Many workflows are composed of short-duration, serial jobs that suffer severe scheduling overheads and queuing delays when submitted via traditional grid access methods. This results in poor performance for applications and poor utilization of resources. We have described how provisioning techniques such as advance reservations, and multi-level scheduling can improve workflow performance by minimizing or eliminating these overheads. We also discussed how emerging IaaS platforms can be combined with multi-level scheduling to create flexible and efficient execution environments for workflow applications.

6. Acknowledgements

This work was supported by the National Science Foundation under grants CCF-0725332 and OCI-0749313.

7. References

[1] Amazon.com, "Elastic Compute Cloud (EC2)"; http://aws.amazon.com/ec2.

[2] E. Byun et al., "Efficient Resource Capacity Estimate of Workflow Applications for Provisioning Resources," *4th IEEE International Conference on e-Science (eScience'08), to appear*, 2008.

[3] S. Callaghan et al., "Reducing Time-to-Solution Using Distributed High-Throughput Mega-Workflows: Experiences from SCEC CyberShake," *4th IEEE International Conference on e-Science (eScience'08), to appear*, 2008.

[4] J.S. Chase et al., "Dynamic virtual clusters in a grid site manager," *12th IEEE International Symposium on High Performance Distributed Computing (HPDC'03)*, 2003, pp. 90-100.

[5] "condor_glidein"; http://www.cs.wisc.edu/condor/glidein.

[6] E. Deelman et al., "GriPhyN and LIGO, building a virtual data Grid for gravitational wave scientists," *11th IEEE International Symposium on High Performance Distributed Computing (HPDC'02)*, 2002, pp. 225-234.

[7] E. Deelman et al., "Managing Large-Scale Workflow Execution from Resource Provisioning to Provenance Tracking: The CyberShake Example," *2nd IEEE International Conference on e-Science and Grid Computing (E-SCIENCE'06)*, 2006.

[8] R.J. Figueiredo et al., "A case for grid computing on virtual machines," *23rd International Conference on Distributed Computing Systems*, 2003, pp. 550-559.

[9] "Flexiscale"; http://www.flexiscale.com.

[10] I. Foster et al., "Virtual Clusters for Grid Communities," *Sixth IEEE International Symposium on Cluster Computing and the Grid (CCGRID'06)*, 2006, pp. 513-520.

[11] I. Foster et al., "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," *International Journal of High Performance Computing Applications*, vol. 15, 2001, pp. 200-222.

[12] J. Frey et al., "Condor-G: A Computation Management Agent for Multi-Institutional Grids," *Cluster Computing*, vol. 5, 2002, pp. 237-246.

[13] "Generic Connection Brokering (GCB)"; http://cs.wisc.edu/condor/gcb.

[14] W. Gentzsch, "Sun Grid Engine: towards creating a compute power grid," *First IEEE/ACM International Symposium on Cluster Computing and the Grid*, 2001, pp. 35-36.

[15] "Glidein Service"; http://www-rcf.usc.edu/~juve/glideinservice.

[16] L. Grit et al., "Virtual Machine Hosting for Networked Clusters: Building the Foundations for "Autonomic" Orchestration," *First International Workshop on Virtualization Technology in Distributed Computing (VTDC'06)*, 2006, pp. 7-7.

[17] R. Huang et al., "Automatic Resource Specification Generation for Resource Selection," *IEEE International Conference on Supercomputing (SC'07)*, 2007.

[18] X. Jiang et al., "SODA: A Service-On-Demand Architecture for Application Service Hosting Utility Platforms," *12th IEEE International Symposium on High Performance Distributed Computing (HPDC'03)*, 2003.

[19] K. Keahey et al., "Virtual workspaces: Achieving quality of service and quality of life in the Grid," *Scientific Programming*, vol. 13, 2005, pp. 265-275.

[20] K. Keahey et al., "Virtual Workspaces for Scientific Applications," *Scientific Discovery through Advanced Computing (SciDAC '07)*, Boston, MA: 2007.

[21] Y. Kee et al., "Enabling personal clusters on demand for batch resources using commodity software," *IEEE International Symposium on Parallel and Distributed Processing (IPDPS'08)*, 2008.

[22] M.J. Litzkow et al., "Condor: A Hunter of Idle Workstations," *8th International Conference on Distributed Computing Systems*, 1988, pp. 104-111.

[23] "Maui Cluster Scheduler"; http://www.supercluster.org/maui.

[24] G. Mehta, "Dynamic Deployment of VO-Specific Condor Scheduler using GT4," *Condor Week*, 2006.

[25] "Montage"; http://montage.ipac.caltech.edu.

[26] D. Nurmi et al., "VARQ: virtual advance reservations for queues," *17th international symposium on High performance distributed computing (HPDC'08)*, 2008.

[27] "PBSPro"; http://www.pbspro.com.

[28] I. Raicu et al., "Falkon: a Fast and Light-weight tasK executiON framework," *Supercomputing (SC'07)*, Reno, Nevada: 2007.

[29] "Science Clouds"; http://workspace.globus.org/clouds.

[30] "SDSC User Portal"; http://portal.sdsc.edu.

[31] I. Sfiligoi, "glideinWMS"; http://home.fnal.gov/~sfiligoi/glideinWMS/.

[32] G. Singh et al., "Optimizing Grid-Based Workflow Execution," *Journal of Grid Computing*, vol. 3, 2005, pp. 201-219.

[33] G. Singh et al., "Performance Impact of Resource Provisioning on Workflows," *USC ISI Technical Report*, 2005.

[34] B. Sotomayor et al., "Overhead Matters: A Model for Virtual Resource Management," *First International Workshop on Virtualization Technology in Distributed Computing (VTDC'06)*, 2006, pp. 5-5.

[35] H. Topcuoglu et al., "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, 2002, pp. 260-274.

[36] E. Walker et al., "Creating Personal Adaptive Clusters for Managing Scientific Jobs in a Distributed Computing Environment," *IEEE Workshop on Challenges of Large Applications in Distributed Environments (CLADE'2006)*, Paris, France: 2006.