Integration of Workflow Partitioning and Resource Provisioning

Weiwei Chen
Information Sciences Institute
University of Southern California
Marina del Rey, CA, USA
wchen@isi.edu

Ewa Deelman
Information Sciences Institute
University of Southern California
Marina del Rey, CA, USA
deelman@isi.edu

Abstract—**The recent increased use of workflow management systems by large scientific collaborations presents the challenge of scheduling large-scale workflows onto distributed resources. This work aims to partition large-scale scientific workflows in conjunction with resources provisioning to reduce the workflow makespan and resource cost.**

Keywords-scientific workflows; workflow partitioning; resource provisioning;

I. Introduction

In recent years, scientific workflows [1] have been widely applied in astronomy [2], seismology [3], genomics [4], etc. A scientific workflow has a sequence of jobs that perform required functionality and it has control or data dependencies between these jobs. The scheduling problem is to map jobs in the workflow onto resources that are often distributed in the wide area. The scheduling is especially challenging for large-scale workflows that have many jobs and data dependencies. Among the solutions, workflow partitioning [5] is an approach to divide a workflow into several sub-workflows and then submit these sub-workflows to diferent execution sites (virtual clusters). Workflow partitioning requires the sub-workflows to be suitable for execution in an execution site in terms of storage constraints, communication cost and computation cost. It is possible that we construct an execution site with all the available resources and do not partition the workflows. However, such a topology may overwhelm the load of some specific nodes, such as the master node or the I/O nodes in the computational cluster. Second, such a system is not reliable and scalable for large-scale workflows and dynamic distributed environments. Partitioning workflows helps to increase the fault tolerance of the overall application since there are multiple virtual clusters available for execution. If a master node in a virtual cluster fails, there are still other virtual clusters available for execution. Third, with more execution sites, the parallelism is increased and the runtime performance of the workflow can be improved.

However, existing work [5] in partitioning is based on the static information of existing execution sites. Nowadays, with the help of resource provisioning [6], we are able to dynamically allocate resources into multiple execution sites or virtual clusters [7] and then execute the sub-workflows on these sites. Our work focuses on the virtual cluster provisioning, which would launch a number of VMs at the same time and build a virtual cluster with them. The VMs may perform different functionalities during the workflow execution. For example, a master node performs workflow task scheduling and monitoring and I/O nodes perform the data management tasks. Such a new approach requires us to consider the workflow partitioning problem and resource provisioning problem together so as to minimize both the workflow execution time (makespan) and the monetary cost of the workflow execution (resource cost).

In this work we propose two methods to implement this integration. The first one is to extend our current work [5] to consider the resource information dynamically. However, it requires the job runtime of a workflow to be well balanced such as the case in the CyberShake workflow [3]. The second one is to combine the resource selection and job scheduling based on Genetic Algorithms (GAs) [8]. Such a method is more adaptive to different workflow structures but it may take longer time to solve the problem because of the complexity of the GAs. Our work aims to compare the two methods.

The goal of our research concerns is to reduce the makespan and the resource cost. The number of resources and the type of VM instances (worker node, master node and I/O node) are the parameters indicating the storage and computational capability of a virtual cluster. We need to consider the topology and structure of a virtual cluster so as to balance the load in different services (scheduling service, data transfer service, etc.) and avoid a bottleneck in any of them. On grids, usually the data transfer service is already available and does not need further configuration. However, how to determine the number of virtual clusters according to the scale of the workflow is still a challenge.

The contributions of our work include a set of algorithms and heuristics to address the issue of integrating workflow partitioning and resource provisioning. We would also build a set of tools to implement the interaction with the Pegasus Workflow Management System (WMS) [9] in the future.

The rest of this paper is organized as follows. In Section II, we present our workflow models. In section III, we describe our research methodology. In section IV, we describe the implementation of our solution. In section V we discuss related work and finally conclude in section VI.

II. Workflow and Workflow Management System

We model workflows as Directed Acyclic Graphs (DAGs), where nodes represent computation jobs and directed edges represent data flow dependencies between nodes. A DAG = (T, E, C, D) [13], where $T = \{t_i ; i = 1,...,n\}$ is a set of jobs, E is a matrix defined on T which specifies operational precedence constraints. That is, $E_{i,i'}=1$ means that t_i must be completed before $t_{i'}$ can start execution, otherwise there is no directed edge from t_i to $t_{i'}$. We assume the completion time of a job t_i on a resource m_j is already known or can be estimated, which is indicated by $C_{i,j}$. D is an $n \times n$ matrix of communication data, where $D_{i,i'}$ is the amount of data required to be transmitted from job t_i to job $t_{i'}$.

In this model, workflow partitioning means to divide the original DAG into several Sub-DAGs. Therefore, DAG = $\cup$ (Sub-DAG$_k$). $k = \{1, \ldots, k_{max}\}$ is the index of sub-workflows. Sub-DAG$_k$ = (T_k, E_k, C_k, D_k), while $T_k \subset T$, $E_k \subset E$, $C_k \subset C$, $D_k \subset D$. The original workflow is replaced by a super workflow Super-DAG that contains sub-workflows as jobs. Super-DAG = (T', E', C', D'), in which T' = {Sub-DAG$_k$;k=1,…,k_{max}} and E', C', and D' are the corresponding matrix defined on T'. The data dependencies between sub-workflows control the order of the execution of sub-workflows. A sub-workflow Sub-DAG$_k$ must be completed before the execution of Sub-DAG$_{k'}$ if there is a data dependency from Sub-DAG$_k$ to Sub-DAG$_{k'}$.

The creation of a super workflow should avoid the cyclic dependencies [5] between sub-workflows and our goal is to reduce the workflow makespan and resource cost for super workflow execution. To estimate the makespan of sub-workflows, similar to [5], we calculate the earliest finish time of the last job in a sub-workflow assuming we use HEFT [22] to schedule the sub-workflow. Using the same approach, we can estimate the makespan of the super workflow (indicated by M). To estimate the resource cost of a workflow execution, we have:

$$Cost = \sum_k (M_k \sum_l n_{lk} p_l) \qquad (1)$$

M_k is the makespan of Sub-DAG$_k$. p_l is the unit price used by Amazon EC2 [15]. n_{lk} is the number of VMs of type l that are used in Sub-DAG$_k$. Normally, nodes that perform different functionalities require different types of VMs.

The aim of our work is to find a set of Sub-DAGs and n_{lk} and minimize the makespan of workflow and the monetary cost of execution. Eq. (2) shows three criteria that are used to represent different preference of users. α and β are weights of makespan and resource cost. M is the makespan of workflow. The cost constraint (budget) and the time constraint (deadline) are specified by the users for workflow execution.

$$Min(\alpha \frac{M}{Deadline} + \beta \frac{Cost}{Budget})$$

$$Min(M), Cost < Budget$$

$$Min(Cost), M < Deadline \qquad (2)$$

The first fitness function in Eq. (2) aims to find a balance between the resource cost and workflow makespan. The second one aims to reduce the workflow makespan while the resource cost is within the budget constraint. The third one aims to reduce the resource cost while the workflow makespan satisfies the deadline.

Our algorithms and heuristics will be implemented in the Pegasus framework. Pegasus WMS is a framework for mapping complex workflows onto distributed resources such as grids and clouds. Internally, Pegasus generates executable workflows from an abstract workflow description and then submits them to DAGMan [10] and Condor [11] for execution. Pegasus has been used to optimize runtime performance of various scientific applications in astronomy [2], biology [4], physics [12], and earthquake science [3] on dedicated clusters, national cyberinfrastructure such as the TeraGrid [13] and the Open Science Grid [14] and clouds such as Amazon EC2 [15] and FutureGrid [16].

III. Research Methodology

In our approach we first design and evaluate our heuristics and algorithms when the resource deployment is static. This step helps us to understand the relationship between partitioning and provisioning and also provides the raw data for the learning process in GA. We then will test the deployments with different number and types of resources. Ideally, the increase of resources would reduce the makespan of workflows but may increase the monetary cost for execution. However, a virtual cluster that consists of different types of resources makes the problem more complicated and challenging. To find a balance between the cost and efficiency, our approach introduces multiple criteria for evaluation.

In the second stage, the fitness function of our GA will consider both the makespan and resource cost. During this period, we will evaluate the performance of our GA with several workflow benchmarks such as Montage [2], CyberShake [3], and Epigenomics [4]. They are chosen because they represent a wide range of application domains and a variety of resource requirements. For example, Montage is I/O intensive, CyberShake is memory intensive and Epigenomics is CPU intensive. The CyberShake workflow is used to calculate Probabilistic Seismic Hazard curves for several geographic sites in the Southern California area. For one typical geographic site, the CyberShake workflow contains 80 partitions. We choose one partition in our experiments that has 24,132 tasks and 58GB of overall data used and consumed. Montage is an astronomy application that is used to construct large image mosaics of the sky. We choose a Montage workflow with a size of 8 degree square of the sky. The workflow has 10,422 tasks and 57GB of overall data. Epigenomics maps short DNA segments collected with high-throughput gene sequencing machines to a previously structured reference genome. The workflow has 1586 tasks and 23GB of overall data. We expect to see significant improvements in terms of resource cost and makespan of workflows if the integration of partitioning and provisioning is well tuned.

In the third stage, we will take the integration with existing tools further. Besides large workflows that have been mentioned before, other forms of workflows, such as workflow of workflows, would be tested to evaluate the scalability of our algorithms. In a Super-workflow containing sub-workflows, a sub-workflow is treated as a job. Dynamic provisioning [18] that automatically adjusts the number and type of VMs can be incorporated with our algorithms.

IV. INTEGRATION OF PARTITIONING AND PROVISIONING

In our work, workflow planning and execution is comprised of three steps: workflow partitioning, resource provisioning and workflow execution.

A. *Workflow Partitioning*

In this step, the partitioner takes the original workflow and the unit cost of resources as input, and outputs multiple sub-workflows (Sub-DAG$_k$) and the resource selection (n_{lk}). The partitioner also instructs the provisioner to deploy resources based on the results of workflow partitioning.

1) *Heuristics*

Heuristics have been proposed and evaluated in our prior work. For example, one heuristic (Heuristic I in [5]) aims to reduce the number of dependencies between sub-workflows as much as possible since such a partitioning can increase parallelism of the execution. In one example of the CyberShake workflow, we have observed that when the resources are evenly distributed on the four execution sites (8CPU slots per site), partitioning our example workflows into 2~3 sub-workflows offers the best balance between computation and communication cost. TABLE I. shows the performance of the Heuristic I with the three workflows. The cumulative disk usage is the sum of the disk usages in all sub-workflows. It increases with the number of execution sites because some data used by jobs in different sub-workflows require replication in different execution sites. We ran each workflow instance 5 times to account for system variability. We aim to extend such experience to the case in which the type and amount of resources are configurable and require optimization based on the partitioning results.

TABLE I. PERFORMANCE OF THE PARTITIONING HEURISTIC

Cumulative Disk Usage (GB)	**Number of Execution Site**			
	4	*3*	*2*	*1*
Montage	74.59	70.2	69.71	56.83
CyberShake	73.81	68.55	63.07	57.6
Epigenomics	30.18	24.32	24.2	23.97
Makespan (second)	**Number of Execution Site**			
	4	*3*	*2*	*1*
Montage	8001.4	7672.6	9226.2	8347.4
CyberShake	9811.6	9917.8	9481	11616.4
Epigenomics	2702.4	2547.2	3200.8	4910.2

2) *Genetic Algorithms*

We extend Genetic Algorithms (GAs) to address the problem of integration of provisioning and partitioning since we can add the information of resources to the chromosome naturally in GAs. GAs start with a set of random solution called initial population and each member of this population is a chromosome that consists of a string of genes. What distinguishes our algorithms from existing GAs is that our chromosome has not only the information about jobs but also the characteristics of resources to be deployed. Figure 1. shows an example chromosome that has five jobs and six VMs. The left part of a chromosome represents the partitioning result of the workflow, which is the index of the sub-workflow that a job belongs to. The right part represents the provisioning of resources, which is the index of the virtual cluster that a resource (VM) belongs to. In this example, Job1 and Job4 are assigned to a sub-workflow (SW1) and other jobs are assigned to SW2. VM1, VM2 and VM3 construct a virtual cluster (VC2) and the remaining VMs construct VC1. A virtual cluster requires at least three nodes (1 master node, 1 I/O node and 1 worker node). Figure 2. shows the two sub-workflows scheduled to these two virtual clusters, although they may not run at the same time.

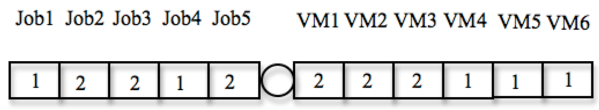

Figure 1. Example of a chromosome.

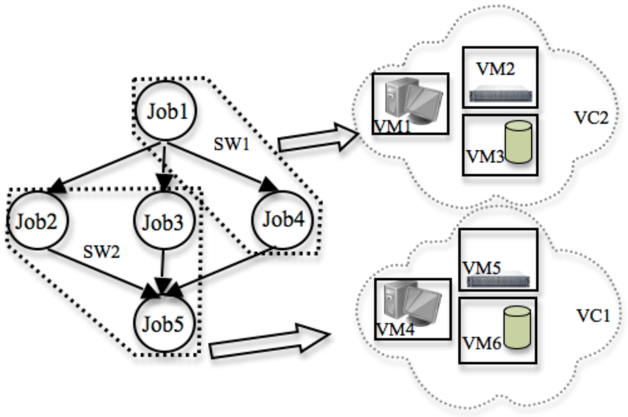

Figure 2. Example of a partitioning and provisioning.

To find a solution, the chromosomes need to go through iteration until some condition(s) is satisfied. A set of chromosomes in each iteration of GA is called a generation. The offspring is created by applying two operators to the current generation: mutation and crossover. Mutation changes randomly some gene values of a chromosome, which means a job may be assigned to another sub-workflow or a resource is assigned to another virtual cluster. Crossover selects two chromosomes of the current population, combines them and creates a new generation.

At the end of each iteration, a fitness function is evaluted to select the best chromosomes and drop others. The fitness function is based on the three criteria in Eq. (2).

B. *Resource Provisioning*

The provisioner launches VM instances and constructs virtual clusters based on the instructions (i.e. n_{lk}) from the partitioner so as to make the provisioning and partitioning consistent. On grids, Corral [19] is used to deploy pilot jobs as placeholders for the execution of application jobs. The required software and services are already installed on the destination nodes. On clouds, Wrangler [20] is used to launch a virtual cluster with and then automatically configure such a virtual cluster with necessary software and services such as file transfer services.

In Wrangler, virtual clusters are specified in a custom XML format. The XML format describes virtual clusters as a collection of several nodes, which correspond to virtual machines. Each node is defined with the characteristics of the virtual machine to be provisioned, such as the VM image to use and the hardware resource type (CPU, memory, disk, etc.).

The provisioner also provides the functionality for managing resources once they are provisioned. In a static way, the provisioner deploys virtual clusters before the execution of workflows and destroys all of them after all of the sub-workflows are completed. In contrast, the dynamic virtual cluster provisioning would destroy virtual clusters once they are no longer needed for any upcoming sub-workflows so as to save costs.

C. *Workflow Execution*

When the resources are available, Pegasus would submit the sub-workflows to the execution sites. Typically, an execution site contains a master node to schedule jobs, one or more I/O nodes to store and manage data and multiple worker nodes to execute workflow tasks. Other options include a data placement service, a logging service, or an access authorization service that may be deployed to different nodes. We assume that a sub-workflow is executed on only one execution site. In each execution of sub-workflows, in a way similar to the normal execution of a workflow, jobs are released when there are no parent jobs that have not completed and then executed at their destination nodes. After the workflow execution, the provisioner destroys virtual clusters and release resources that were deployed.

V. Related Work

There have been a considerable amount of work trying to solve workflow-mapping problem using DAG scheduling heuristics such as HEFT [22], Min-Min [21], MaxMin [24], MCT [24], etc. Sonmez [25] extended them to multiple workflows in multi-cluster grids. Duan [23], Wieczorek [26] have discussed the scheduling and partitioning scientific workflows in dynamic grids. Our work aims to partition workflows with resource provisioning.

In our prior work, we have developed a three-phrase scheduling approach integrated with the Pegasus WMS to partition, estimate, and schedule workflows onto distributed resources. Three heuristics were proposed and evaluated to partition workflows respecting storage constraints and internal job parallelism. Three methods were used to estimate and compare runtime of sub-workflows and then these sub-workflows were scheduled based on two commonly used algorithms. In this work, we aim to incorporate the resource provisioning into the workflow partitioning, which would influence the partitioning results as well.

The use of graph partitioning for the workflow DAG has been discussed in [27][28][29]. Our work aims to apply these solutions to the problem of resource provisioning and workflow partitioning. Papers [30][31] proposed the use of graph partitioning for partition the resources of a distributed system, but not the workflow DAG, which means the resources are provisioned into different execution sites but the workflows are not partitioned at all.

VI. Conclusion and Future Work

Workflows partitioning and resource provisioning remains a challenging problem. In this paper we showed that genetic algorithms and heuristics could be a good candidate to solve this problem.

In this paper we wanted to discover an efficient workflow scheduling algorithm by focusing on the integration of resource provisioning and workflow partitioning. In the future, we aim to investigate more workflows and evaluate more algorithms besides heuristics and GAs.

References

[1] I. J. Taylor, E. Deelman, et al. Workflows for e-Science. Scientific Workflows for Grids. Springer, 2007.

[2] G. B. Berriman, et al., "Montage: a grid-enabled engine for delivering custom science-grade mosaics on demand," in Proceedings of SPIE vol. 5493, ed, 2004, pp. 221-232.

[3] P. Maechling, E. Deelman, et al. SCEC CyberShake Workflows—Automating Probabilistic Seismic Hazard Analysis Calculations. Workflows for e-Science. Scientific Workflows for Grids. Springer, 2007.

[4] USC Epigenome Center, http://epigenome.usc.edu.

[5] W. Chen, E. Deelman, Partitioning and Scheduling Workflows across Multiple Sites with Storage Constraints, Workshop on Scheduling for Parallel Computing, 9th Intl. Conf. on Parallel Processing and Applied Mathmatics, Sep 2011.

[6] Gideon Juve, Ewa Deelman, Karan Vahi, Gaurang Mehta, Experiences with Resource Provisioning for Scientific Workflows Using Corral, Scientific Programming, 18:2, pp. 77-92, April 2010.

[7] Nicolae, B., Bresnahan, J., Keahey, K., Antoniu, G, Going Back and Forth: Efficient Multideployment and Multisnapshotting on Clouds, HPDC 2011. San Jose, CA. June 2011.

[8] Mohammad Daoud and Nawwaf Kharma. GATS 1.0: a novel GA-based scheduling algorithm for task scheduling on heterogeneous processor nets. In Proceedings of the 2005 conference on Genetic and evolutionary computation (GECCO '05), Hans-Georg Beyer (Ed.).

[9] E. Deelman, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, S. Patil, M.-H. Su, K. Vahi, and M. Livny. Pegasus: Mapping scientific workflows onto the Grid. Lecture Notes in Computer Science: Grid Computing, pages 11–20, 2004.

[10] Peter Couvares, Tevik Kosar, Alain Roy, Jeff Weber and Kent Wenger, "Workflow in Condor", in *In Workflows for e- Science*, Editors: I.Taylor, E.Deelman, D.Gannon, M.Shields, Springer Press, January 2007 (ISBN: 1-84628-519-4)

[11] J. Frey, T. Tannenbaum, M. Livny, I. Foster, and S. Tuecke, "Condor-G: A Computation Management Agent for Multi-Institutional Grids," Cluster Computing, vol. 5, pp. 237-246 2002.

[12] E. Deelman, J. Blythe, Y. Gil, and C. Kesselman, "Workflow Management in GriPhyN," in *Grid Resource Management: State of the Art and Future Trends*, J. Nabrzyski, J. M. Schopf, and J. Weglarz, Eds.: Springer, 2003.

[13] C. Catlett, "The philosophy of TeraGrid: building an open, extensible, distributed TeraScale facility," presented at Cluster Computing and the Grid 2nd IEEE/ACM International Symposium CCGRID2002, 2002.

[14] The Open Science Grid Consortium, http://www.opensciencegrid.org.

[15] Amazon.com, “Elastic Compute Cloud (EC2)”; http://aws.amazon.com/ec2.

[16] FutureGrid: https://portal.futuregrid.org/

[17] Infrared Processing and Analysis Center, http://www.ipac.caltech.edu/

[18] J.S. Chase, et. al., “Dynamic virtual clusters in a grid site manager,” Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing (HPDC03), 2003, pp. 90-100

[19] Gideon Juve, Ewa Deelman, Karan Vahi, Gaurang Mehta, Experiences with Resource Provisioning for Scientific Workflows Using Corral, Scientific Programming, 18:2, pp. 77-92, April 2010.

[20] Gideon Juve and Ewa Deelman, Wrangler: Virtual Cluster Provisioning for the Cloud, short paper, Proceedings of the 20th International Symposium on High Performance Distributed Computing (HPDC'11), 2011.

[21] J. Blythe, S. Jain, E. Deelman, et al., Task Scheduling Strategies for Workfow-Based Applications in Grids. CCGrid 2005.

[22] H. Topcuoglu, S. Hariri, et al. Performance-Effective and Low-Complexity Task Scheduling for Heterrogeneous Computing, IEEE Transactions on Parallel and Distributed System, Vol 13, No. 3, Mar 2002.

[23] R. Duan, R. Prodan, et al., Run-time Optimisation of Grid Workflow Applications, 7th IEEE/ACM International Conference on Grid Computing, Sep 2005.

[24] T. D. Braun, H. J. Siegel, et al., A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems. Journal of Parallel and Distributed Computing, 61, pp. 810-837, 2001.

[25] OO. Sonmez. Application-Oriented Scheduling in Multicluster Grids. http://www.pds.ewi.tudelft.nl/~sonmez/research.htm, June 2010.

[26] M. Wieczorek, R. Prodan, et al. Scheduling of scientific workflows in the ASKALON grid environment, SIGMOND Record, Vol 34 Issue 3, Sep 2005.

[27] F. Dong and S. Akl, “Two-phase computation and data scheduling algorithms for workflows in the grid,” in Parallel Processing, 2007. ICPP 2007. International Conference on. IEEE, 2007, pp. 66–66.

[28] S. Kalayci, G. Dasgupta, L. Fong, O. Ezenwoye, and S. Sadjadi, “Distributed and adaptive execution of condor dagman workflows.”

[29] O. Sonmez, N. Yigitbasi, S. Abrishami, A. Iosup, and D. Epema, “Performance analysis of dynamic workflow scheduling in multicluster grids,” in Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing. ACM, 2010, pp. 49–60.

[30] S. Kumar, S. Das, and R. Biswas, “Graph partitioning for parallel applications in heterogeneous grid environments,” in Parallel and Distributed Processing Symposium., Proceedings International, IPDPS 2002, Abstracts and CD-ROM. IEEE, 2002, pp. 66–72.

[31] C. LIN, C. SHIH, and C. HSU, “Adaptive dynamic scheduling algorithms for mapping ongoing m-tasks to pr 2 grid,” Journal of information science and engineering, vol. 26, pp. 2107–2125, 2010.