ISSN: 2096-3246
Volume 55, Issue 01, Jan, 2023

Data Needs Analysis for Scheduling Kepler Cloud-Based Workflow

N.SRINADH REDDY[1],RAJA BHARGAV[2],P.FIRIDOZ KHAN[3],
ASSOCIATE PROFESSOR[1],[2],ASSISTANT PROFESSOR[3],
DEPARTMENT OF CSE
PBR VISVODAYA INSTITUTE OF TECHNOLOGY AND SCIENCE::KAVALI

Keywords:
Kepler, cloud workflow, workflow scheduling, scheduling information.

Abstract

Scientists have relied on the Kepler scientific workflow system to help them automate experiments across many different fields using distributed computing platforms. An assigned director oversees the execution of a process in Kepler. Users must still choose the computing resources that will run the workflow's tasks. A workflow scheduler that can allocate workflow tasks to resources for execution is needed to further reduce the technical effort required by scientists. We evaluate numerous cloud workflow scheduling methods to determine what data must be exposed for a scheduler to successfully plan for the execution of a Kepler process in the cloud. We explain the value by discussing the advantages of each different kind of data about workflow jobs, cloud resources, and cloud service providers.

Introduction

Scientific workflow management systems, such as Kepler [1], have been used for facilitating eScience by providing tools to orchestrate scientific computations to be executed automatically. These systems can execute workflows on local resources, or on distributed environments such as the grid, which provides supercomputing power necessitated by intensive computations in scientific applications [2]. As the cloud computing emerges as a new computing paradigm, it has become possible to execute scientific workflows on cloud resources [3], which are allocated on demand reducing hardware investment. Because of several benefits offered by the cloud, more research effort has been directed toward the execution of workflows in cloud computing environment. An essential mechanism for executing a workflow efficiently in the cloud is a scheduler, which decides on the mapping of the tasks (or actors in Kepler) in the workflow to cloud resources. To control Kepler workflow execution, several “directors” are available for users to choose for their workflows [4]. However, the directors in Kepler do not perform scheduling of actors to compute resources; users have to specify the resources on which the actors are to be executed. In order forKepler to relieve scientists from technical complication, a scheduler may be necessary. Toward a development of this scheduler, in this paper we identify the types of information that should be supplied by the Kepler environment to its scheduler based on existing cloud workflow scheduling techniques.

Unlike scheduling in the grid, workflow scheduling in the cloud context needs to address new issues specific to cloud computing. For example, instead of trying to finish a workflow execution as soon as possible, cloud workflow schedulers also have to control the cost incurred by the execution in the cloud. The resources in the cloud, being virtual machines in this case, are more homogeneous and are dependent upon the request from users (or from the schedulers) whereas grid resources are more heterogeneous and are not controlled by users [3]. Such issues shift the way in which scheduling techniques are designed, which therefore require additional information for scheduling. This paper explores a number of cloud workflow scheduling techniques that utilize information specific to cloud computing environment in their scheduling processes. However, as cloud computing and cloud providers evolve, some techniques (and its required information) may no longer be as useful. Thus, we also discuss and justify each of the information identified whether it is of important in the current cloud computing context. The structure of this paper is as follows. Section 2 presents a review of cloud workflow scheduling techniques. Section 3 discusses and summarizes the use of such information to address issues specific to cloud environment. Section 4 then concludes the paper and points toward our future work.

Cloud Workflow Scheduling Techniques

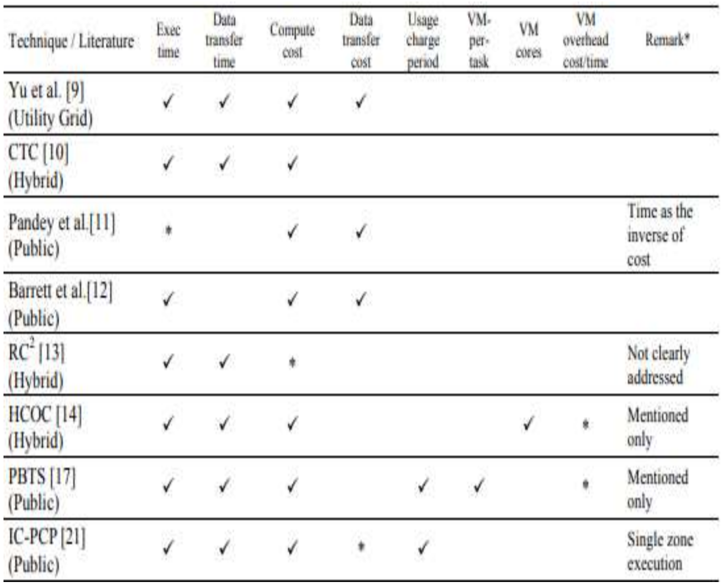
Before the emergence of cloud computing, scientific workflows had usually been executed in the grid and extensive effort had been put into grid workflow scheduling resulting in several techniques over many years. These techniques, however, may not be applicable to scheduling workflows in the cloud because of the differences between the grid and the cloud context. Thus, researchers have diverted their effort to cloud workflow scheduling. Scheduling techniques have been developed in response to the evolving cloud technology and applications that runs in the cloud. More information regarding workflows and cloud resources has become necessary in a scheduling process. Workflow scheduling in the grid usually relies on three common metrics, which are the execution time of tasks, the wait time on grid resources, and the time required for file staging or data transfer time [5]. Additional metrics or information may be utilized by some techniques to address specific issues such as resource competition [6], fault tolerance [7], and reliability [8]. With cloud computing, compute resources or virtual machines can be requested at runtime while users are charged for the use of such resources. Therefore, the three common metrics are no longer sufficient as the total execution cost has become a more important metric. This section explores several cloud workflow scheduling techniques to identify necessary scheduling information specific to cloud computing. We present these techniques in chronological order so that it may be possible to illustrate the evolution of the researches in this area. Although many grid workflow scheduling techniques focus on minimizing makes pan, a few techniques also address execution cost in utility grid. For example, Yu et al. [9] proposed a technique that partitions a workflow into branches based on synchronization tasks and distributed the global deadline among such partitions. The cost is then minimized for each partition while trying to preserve the deadline of the partition. This early technique considers the usage cost of grid resources and network cost for transferring data. In 2009, Liu [10] proposed, among a number of algorithms in his thesis, the “CTC” algorithm. This algorithm addresses two objectives that are to minimize execution cost within user-defined deadline and to minimize makes pan within user-defined budget. As one of earlier techniques, this algorithm considers execution time, data transfer time, and compute cost to allocate cheaper cloudresources to workflow tasks. The estimated total cost and makes pan are available for user to make any adjustment of deadline and budget during execution and the algorithm reschedules as necessary. Pandey et al. [11] employ particle swarm optimization (PSO) to schedule workflows on cloud compute resources. PSO is used to generate a schedule for tasks that are ready to execute in each scheduling round.

The optimization of the PSO considers the compute cost and the data communication cost to minimize total execution cost. The technique tries to avoid communication cost when data files are larger, and is able to balance the load on all resources based on their cost. The literature does not explicitly consider task execution time and data transfer time, and thus does not address time constraint. However, the aspect of time is expressed as the inverse proportion to the compute cost. Another technique that employed a meta-heuristic was proposed by Barrett et al. [12]. This technique uses genetic algorithm to generate multiple schedules based on makes pan, compute cost, and data transfer cost. Then among these schedules, a Markov Decision Process chooses the solution based on resource loads and the hour of the day. An algorithm for scheduling tasks in hybrid cloud, called “RC2 ” was proposed by Lee and Zumaya [13] to achieve reliable completion. An initial schedule is first calculated based on private cloud (or locally owned resources) to minimize cloud resource usage. If a delay occurs at a local resource, tasks that have been scheduled to that resource may be rescheduled to cloud resources to compensate the delay. The RC2 algorithm considers task execution time and data transfer time to calculate makes pan in its scheduling process. However, this algorithm does not explicitly consider computed cost. In 2011, Bettencourt and Madeira [14] proposed the “HCOC” algorithm to schedule cloud workflows within deadline while minimizing compute cost. The algorithm assumes hybrid cloud model consisting of a private cloud of heterogeneous resources and a public cloud of (unlimited) resources. The scheduling process starts by using the grid workflow scheduling algorithm called “PCH” [15] to generate an initial plan for the private cloud based on execution time and data transfer. If the initial schedule is expected to violate the deadline, rescheduling is triggered to select and reassign tasks to public cloud resources. The selection of public cloud resources is based on cost, performance and number of cores on cloud resources. The authors also suggested later that it is necessary to consider an overhead cost incurred by a cloud resource being idle while waiting for input data to be transferred [16].

However, this had not been included in the HCOC algorithm. In addition to task execution time, data transfer time, and compute cost that are used in the techniques described so far, the “PBTS” algorithm proposed by Byun et al. [17] begins to consider other aspects in the cloud. To addresses tasks implementing the MapReduce [18] programming model, the algorithm assumes that a task may requires varying numbers of virtual machines. In order to minimize the cost of cloud resources while maintaining deadline, this algorithm, as the extension of the authors' earlier algorithm called "BTS" [19], divides a workflow schedule into partitions based on the charge period of cloud resource. For example, users are charged based on hourly rates for using Amazon EC2 services [20]. A workflow is scheduled so that the virtual machine hours are minimized by trying to fit tasks in the charge periods. This consideration is necessary because a virtual machine may be utilized for only a short time and then left idle while users have to pay the price for the whole period, resulting in a higher total cost [17].

It is also mentioned in this work that the overhead for starting virtual machines should be considered. However, this issue is not addressed by the PBTS algorithm. The most recent technique in our review is proposed by Abishai et al. [21], appearing in 2013. Similar to the PBTS [17], this technique, consisting of the "IC-PCP" and the "IC-PCPD2" algorithms, also considers usage charge period in its scheduling process and tries to utilize the remaining time of each period as much as possible. The IC-PCP algorithm iteratively determines a critical path and its deadline then assigns all tasks in the path to a machine that can execute them within the deadline. The IC-PCPD2 algorithm, instead, distributes the workflow deadline to each task and assigns each task to a virtual machine that can execute the task within its deadline. This technique assumes that theexecution of a workflow takes place in a single cloud availability zone thus ignoring data transfer cost between virtual machines. From the cloud workflow scheduling techniques described so far, the information that has been considered by each technique can be summarized in Table 1 with the cloud execution model specified below each technique. The three most essential metrics, which are used in most of the techniques described in this paper, are task execution time, compute cost, and data transfer time because they are required for estimating cost and makes pan. In the next section, we discuss the applicability of the other five in the current cloud computing context.

Table 1: Summary of the information utilized by cloud workflow scheduling techniques



Technique / Literature	Exec time	Data transfer time	Compute cost	Data transfer cost	Usage charge period	VM-per-task	VM cores	VM overhead cost/time	Remark*
Yu et al. [9] (Utility Grid)	✓	✓	✓	✓					
CTC [10] (Hybrid)	✓	✓	✓						
Pandey et al.[11] (Public)	*		✓	✓					Time as the inverse of cost
Barrett et al.[12] (Public)	✓		✓	✓					
RC^2 [13] (Hybrid)	✓	✓	*						Not clearly addressed
HCOC [14] (Hybrid)	✓	✓	✓				✓	*	Mentioned only
PBTS [17] (Public)	✓	✓	✓		✓	✓		*	Mentioned only
IC-PCP [21] (Public)	✓	✓	✓	*	✓				Single zone execution

Discussion on Information Utilized in Workflow Scheduling

The inclusion of the cost for transferring data in and out of a cloud has received less attention than the three-essential metrics. The inclusion of this information may depend on the assumption made by each technique. Infrastructure-as-a-Service cloud providers such as Windows Azure [22] and Amazon EC2 [20] charge minimal cost for transferring data to and within their cloud centers. If it is assumed that a workflow execution takes place only in a single cloud, this cost will be incurred mostly by retrieving output data out of a cloud center at the end of the execution. If a workflow is executed in a hybrid cloud, then the cost is also dependent on the data dependencies between tasks because data may need to be transferred from cloud center to private resources during execution and vice versa. In a scenario of "intercloud" [23], where a workflow is executed using resources from multiple cloud providers, the cost incurred by transferring data across providers become more complicated due to different pricings. Nevertheless, this cost is also subject to the size of data being transferred throughout an execution. Thus, for a system supporting mainly data-intensive workflows, it is necessary to consider data transfer cost in the scheduling mechanism. The more advanced techniques described in this paper (PBTS and IC-PCP) consider usage charge period of virtual machines. This consideration is significant if a workflow is composed mostly of tasks whose execution times are shorter than the charge period. Otherwise, the allocated virtual machines would be underutilized and users would have to pay higher for idle time [17, 21]. However, as of2013, Windows Azure and Google Compute Engine charge their virtual machines per minute.

The shorter charge period renders the consideration of charge period less beneficial [21], which may not justify the complexity of the scheduling algorithms and implementations. Nonetheless, some cloud providers such as Amazon EC2 still charges their service based on hourly rate. The consideration of charge period may still be useful until most cloud providers adjust to shorter charge period. A shorter charge period makes it easier to allocate a virtual machine for a short period of time for a smaller task at reduced cost. However, doing so may lead to undesirable overhead. As mentioned in [3, 16, 17], the overhead of starting up a virtual machine should be considered. A virtual machine, though assuming its image is stored in the cloud, takes time before it is ready to execute a task (such as booting OS) [3]. Also, the machine may have to idly wait for input data to be transferred before the execution can start [16]. These issues introduce additional cost and time in an execution. A scheduler needs to consider this overhead to decide whether to allocate a new virtual machine (to meet a deadline) and/or to terminate one (to reduce cost when it is no longer required). As various instance types of virtual machine are made available by cloud providers, a scheduler using algorithm such as HCOC can select an instance type based on number of cores to execute tasks in different workflow paths on the same virtual machine. This helps minimizing data transfers between tasks and reducing cost per core when a workflow has a high level of parallelism [14]. As for tasks implementing MapReduce model, the consideration shifts from the number of cores to the number of virtual machines. In most of the existing cloud and grid workflow scheduling techniques, a task is usually assumed to be executed by only a single resource to reduce scheduling complexity. Since MapReduce applications are usually executed on multiple worker machines in parallel, neglecting this virtual machine requirement may lead to degraded performance. This information is thus necessary, as shown in the PBTS algorithm [17], which allows a task to specify either static or adjustable number of virtual machines.

Information Requirement for Cloud Workflow Scheduler

In summary, for an implementation of a workflow scheduler in the Kepler system, at least the three essential metrics namely execution time, data transfer time, and compute cost need to be provided in order to estimate cost and makes pan. The execution time can also be expressed as virtual machine performance (e.g., MIPS) and the size of task (e.g., millions of instructions) [16]. Alternatively, the scheduler can maintain a record of the execution time of each task on each virtual machine within itself (or as a part of provenance) to later determine expected execution times. The data transfer time can be expressed as the size of data to be transferred and the bandwidth of network links (which, in the simplest manner, can be recorded within the scheduler to estimate the expected transfer time) [6]. The data transfer costs of transferring data to and from each cloud provider in each availability zone (or region) should also be supplied to support scheduling of data-extensive workflows. The number of cores specified in each virtual machine instance type should be exposed to the scheduler to exploit workflow parallelism. The number of virtual machines required by each task is required to support distributed programming paradigm. This number may be specified (by users) as a parameter attached to the actor representing the task. The overhead for starting virtual machine of different types, although has not been clearly addressed in any work presented here, should be tracked to justify an allocation and a termination [3]. This tracking could be implemented inside the scheduler itself. Lastly, for the usage charge period, it is our opinion that it may become unnecessary. As more cloud providers shorten their charge periods, its usefulness diminishes and thus may not justify the overhead of complex scheduling process and implementation.

Conclusion and Future Work

In this research, we investigate alternative methods for scheduling cloud-based workflows and extrapolate the data that the Kepler system's environment should provide for deploying such a scheduler. The value of various forms of data in the modern era of cloud computing is examined. Scheduling actors in a workflow onto computational resources is handled by "Nimrud Director" [26], which is based on our prior work developing a prototype scheduler for the tool "Nimrud/K" [6, 24, 25], which is built on top of the Kepler system. A cloud-based process scheduler may be implemented in the Kepler system, and this may be a viable alternative. However, since it was originally designed for a grid environment, the prototype scheduler overlooked this revolutionary feature of cloud computing. In addition, the Kepler system may not provide all of the statistics outlined in this work. This includes compute cost, data transmission cost, and the number of cores in each virtual machine. In practice, this may need the inclusion of a supplementary component. There are numerous methods that make assumptions about workflow and cloud execution settings beyond the information and concerns mentioned in this research, including task requirement and user interaction [10]. As a follow-up, we're doing a thorough literature review to compile a taxonomy that may guide future innovation in cloud workflow scheduling methods and tools.

References

[1] B. Ludäscher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger, M. Jones, E. A. Lee, J. Tao, and Y. Zhao, "Scientific workflow management and the Kepler system," Concurr. Comput. : Pract. Exper., vol. 18, pp. 1039-1065, 2006.

ISSN: 2096-3246
Volume 55, Issue 01, Jan, 2023

[2] E. Deelman, D. Gannon, M. Shields, and I. Taylor, "Workflows and e-Science: An overview of workflow system features and capabilities," Future Generation Computer Systems, vol. 25, pp. 528-540, 2009.

[3] G. Mateescu, W. Gentzsch and C. J. Ribbens, "Hybrid Computing—Where HPC meets grid and Cloud Computing," Future Generation Computer Systems, vol. 27, pp. 440-453, 2011.

[4] Kepler User Manual. Available: https://kepler-project.org/users/documentation Accessed January, 2014.

[5] J. Yu, R. Buyya and K. Ramamohanarao, "Workflow Scheduling Algorithms for Grid Computing," in Metaheuristics for Scheduling in Distributed Computing Environments, ed, 2008, pp. 173-214.

[6] S. Smanchat, M. Indrawan, S. Ling, C. Enticott, and D. Abramson, "Scheduling parameter sweep workflow in the Grid based on resource competition," Future Generation Computer Systems, vol. 29, pp. 1164-1183, 2013.

[7] K. Plankensteiner, R. Prodan and T. Fahringer, "A new fault tolerance heuristic for scientific workflows in highly distributed environments based on resubmission impact," in Proceedings of the 5th IEEE International Conference on e-Science (e-Science '09), Oxford, UK, 2009, pp. 313-320.

[8] Y. Tao, H. Jin, S. Wu, X. Shi, and L. Shi, "Dependable Grid Workflow Scheduling Based on Resource Availability," J. Grid Comput., vol. 11, pp. 47-61, 2013.

[9] J. Yu, R. Buyya and C. K. Tham, "Cost-based scheduling of scientific workflow applications on utility grids," in Proceedings of the First International Conference on e-Science and Grid Computing, 2005, pp. 140-147.

[10] K. Liu, "Scheduling algorithms for instance-intensive cloud workflows," PhD Thesis, Faculty of Information and Communication Technologies, Swinburne University of Technology, 2009.

[11] S. Pandey, L. Wu, S. M. Guru, and R. Buyya, "A Particle Swarm Optimization-Based Heuristic for Scheduling Workflow Applications in Cloud Computing Environments," in Proceedings of the 24th IEEE International Conference on Advanced Information Networking and Applications (AINA), 2010, pp. 400-407.

[12] E. Barrett, E. Howley and J. Duggan, "A Learning Architecture for Scheduling Workflow Applications in the Cloud," in Proceedings of the 9th IEEE European Conference on Web Services (ECOWS), 2011, pp. 83-90.

[13] Y. C. Lee and A. Y. Zomaya, "Rescheduling for reliable job completion with the support of clouds," Future Generation Computer Systems, vol. 26, pp. 1192-1199, 2010.

[14] L. F. Bittencourt and E. R. M. Madeira, "HCOC: A Cost Optimization Algorithm for Workflow Scheduling in Hybrid Clouds," Journal of Internet Services and Applications, vol. 2, pp. 207-227, 2011.

[15] L. F. Bittencourt and E. R. M. Madeira, "A performance-oriented adaptive scheduler for dependent tasks on grids," Concurr. Comput. : Pract. Exper., vol. 20, pp. 1029-1049, 2008.

[16] L. F. Bittencourt, E. R. M. Madeira and N. L. S. d. Fonseca, "Scheduling in hybrid clouds," IEEE Communications Magazine, vol. 50, pp. 42-47, 2012.

[17] E. Byun, Y. Kee, J. Kim, and S. Maeng, "Cost optimized provisioning of elastic resources for application workflows," Future Generation Computer Systems, vol. 27, pp. 1011-1026, 2011.

[18] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," Commun. ACM, vol. 51, pp. 107-113, 2008.

[19] E. Byun, Y. Kee, J. Kim, E. Deelman, and S. Maeng, "BTS: Resource capacity estimate for time-targeted science workflows," Journal of Parallel and Distributed Computing, vol. 71, pp. 848-862, 2011.

[20] Amazon EC2. Available: http://aws.amazon.com/ec2/ Accessed November, 2013.

[21] S. Abrishami, M. Naghibzadeh and D. H. J. Epema, "Deadline-constrained workflow scheduling algorithms for Infrastructure as a Service Clouds," Future Generation Computer Systems, vol. 29, pp. 158-169, 2013.

[22] Windows Azure: Infrastructure Services. Available: http://www.windowsazure.com/enus/solutions/infrastructure/ Accessed November, 2013.

[23] R. Buyya, R. Ranjan and R. N. Calheiros, "InterCloud: Utility-Oriented Federation of Cloud Computing Environments for Scaling of Application Services," in Algorithms and Architectures for Parallel Processing. vol. 6081, C.-H. Hsu, et al., Eds., ed: Springer Berlin Heidelberg, 2010, pp. 13-31.

[24] D. Abramson, C. Enticott and I. Altintas, "Nimrod/K: towards massively parallel dynamic grid workflows," in Proceedings of the 2008 ACM/IEEE conference on Supercomputing, Austin, Texas, 2008.