A Study on Cloud Cost Efficiency by Exploiting Idle Billing Period Fractions

Altino M. Sampaio
CIICESI, Escola Superior de Tecnologia e Gestão de Felgueiras, Instituto Politécnico do Porto
Felgueiras, Portugal
Email: ams@estgf.ipp.pt

Jorge G. Barbosa
LIACC, Departamento de Engenharia Informática
Faculdade de Engenharia, Universidade do Porto
Porto, Portugal
Email: jbarbosa@fe.up.pt

***Abstract*—In most of the current commercial Clouds, resources are billed based on a time interval equal to one hour, as is the case of virtual machine (VM) instances on Amazon EC2. Such time interval is usually long, and yet the user has to pay for the whole last hour, even if he/she has only used a fraction of it, contradicting the pay-as-you-go model of Clouds.**

In this paper, we analyse the advantages of adopting alternative scheduling policies that exploit idle last time intervals, in terms of service cost to Cloud users and operating costs to Cloud providers. Using a real-life astronomy workflow application, constrained by user-defined Deadline and Budget quality of service (QoS) parameters, a set of online state-of-the-art-based scheduling algorithms try different execution and resource provisioning plans. Our results show that exploitation of partially idle last time intervals can reduce the cost of service to the end user, and augments providers competitiveness up to 21.6% through energy efficiency improvement and consequent lowering of operational costs.

***Keywords*-Cloud computing; workflow scheduling; service and operating cost analysis.**

I. Introduction

Workflows are commonly used for modeling a wide range of applications in distributed systems [1]. Such applications frequently encompass various domains including science, engineering, consumer, and business. Typically, a workflow is described by a graph that consist of a set of nodes (or vertices) and a set of edges, where nodes represent computational tasks or data transfers, and edges represent control and data dependencies [2]. In particular, many of workflow applications fall in the category of Directed Acyclic Graph (DAG) [3], where the edges represent the temporal relations between the tasks.

One of the most challenging problems in distributed systems is workflow scheduling, which refers to the problem of spatial and temporal mapping of workflow tasks onto resources in order to satisfy some or multiple performance criterion. Since task scheduling is a well-known NP-complete problem [4], several heuristic algorithms have been proposed for the scheduling of workflows onto distributed systems like Grids and Clouds [5], [6], [7]. These scheduling algorithms follow some cost model, in order to optimize specific objectives, such as execution time, economic cost, and data quality.

Two of the most relevant user concerns regards to cost and time. In this context, a frequent approach in the process of executing users' workflows relates to the strategies for cost- and deadline-constrained schedules. Such approach has been rapidly growing and it is supported by generally adopted utility computing model in distributed systems, in which users consume services and resources when they need them and are charged based on the number of billing time intervals that they have used, even if they have not completely used the last time interval [8].

In this paper, we explore the opportunity to reuse such unused time from the last time intervals, by assigning them to other users so they can execute their workflow tasks. The objective is to reduce the cost to execute workflows, while assuring that the deadline-constraint is observed. Such an approach can translate in the reduction of the cost of executing workflow tasks from the user's perspective, and the ability to improve resource utilization and decrease of operational costs from the provider's viewpoint. Thus, by putting a higher priority on cost efficiency than cost effectiveness might become more beneficial to both the user and the provider.

The rest of the paper is organised as follows. Section II discusses related work. Section III introduces the system model, the application model and the scheduling objectives. Section IV presents the tested scenario by defining the simulator, experimental workflows, the algorithms, and performance metrics. Section V presents the discusses the obtained results. Conclusions are presented in Section VI.

II. Related Work

There has been relevant research on scheduling multiple workflows in distributed systems. These scheduling strategies try to find the best workflow to resources mapping, in order to satisfy users' budget and deadline specifications. Alkhanak et al. [5] provided an extensive study about proposed cost-aware workflow scheduling strategies. Based on extensive literature review, the authors identified cost-aware workflow scheduling challenges related to robustness, flexibility, and approach design of scheduling approaches. All the analysed strategies contribute with interesting scheduling approaches, exploring different characteristics and aiming at

978-1-5090-3684-4/16 $31.00 © 2016 IEEE
DOI 10.1109/IC2EW.2016.41

IEEE computer society

optimizing specific objectives. However, they do not aim at exploring unused time intervals, nor were tested in such scenario.

Arabnejad and Barbosa [6] have proposed an online multi-workflow deadline- and budget-constrained scheduling algorithm (MW-DBS), for scheduling concurrent workflow applications. In this sense, the proposed scheduling algorithm considers both the time and cost as constraints and do not perform optimization. Authors argue that their multiple workflow scheduling approach is the first multiple online workflow scheduling algorithm that simultaneously considers user's budget and deadline constraints for concurrent workflow scheduling in heterogeneous computing systems. The algorithm works in two steps: first, a ready task from each workflow is selected and a priority based on individual deadline is assigned; then, a suitable resource to execute the current task that satisfies the QoS parameters of the workflow where the task belongs, is determined at a second step. Based on simulation with well-known workflow structures, authors showed that MW-DBS is able to maximize the number of workflows that can execute bounded by the user specified deadline and budget. Despite the promising results, authors assumed that the cost imputed to the workflow execution is only the effective used time.

Aiming at decreasing the cost of executing high-performance computing (HPC) applications in cloud environments, Niu and Tang [7] proposed a Semi-Elastic Cluster (SEC) computing model to reserve and dynamically resize a virtual cloud-based cluster. The idea is to aggregate the workloads from concurrent users and enjoy from deep discount to heavy users through reserved instances, a policy that has been done by some cloud resource providers such as Amazon. Unfortunately, previous studies have shown that Amazon EC2 instances incur a measurable boot time on the on-demand option (by several minutes in the worst-case) [9].

Other projects such as Amazon EC2 spot instances (i.e., virtual execution environments) [10] explore cost-saving approaches, by selling the idle time in EC2 data centers. Despite the low price offered, the instances are not reliable since tasks are terminated immediately if the current spot price exceeds the initially offered price. In this case, reliability approaches are needed [9] so work done so far is not lost. Nowadays, some providers support short time intervals, such as CloudSigma offering time interval of five minutes [11], but the advantages of such granularity is unclear and the provider seems to have abandoned such pricing model at the time of this publication.

Unlike related works, in this paper we analyse the advantages to state-of-the-art workflow scheduling algorithms in exploiting the unused last time intervals, maintaining the usual pricing model, so the cost of executing tasks can eventually be decreased, captivating users and making providers more competitive. The strategy is to enable users to reuse others users' idle last time intervals fractions.

III. System Overview

A. Resource Model

We assume a resource model typical in distributed systems such as Clouds, where VM instances may be provisioned on-demand and are charged per time unit [8]. Typically, the price is defined in such a way that the VM instances with the most powerful processors have the highest cost and the VMs with less powerful processors have the lowest cost. Current commercial Clouds follow the pay-as-you-go pricing model, charging users based on the number of the time intervals that they have consumed. It is a common practice in Cloud environments to charge per hour, such as in Amazon EC2, which usually turns out to be a long period [12]. A VM instance is terminated at the end of its charging period, which means that an idle VM remains available during the whole billing cycle. As a consequence, even if only a small fraction of the last time interval is utilized, the user is fully charged. In this regarding, if a task takes s time units to process in a resource that costs $\$d$ per time unit, then the cost of executing the task in that resource is $s \times \$d$.

Usually, providers deploy a limited number of heterogeneous VM types, each of which with a specific amount of CPU and memory (i.e., resource characteristics are constant during their lifetime). A workflow task executes in a single VM instance. We also assume that a submitted task has exclusive access to a VM instance while it is executing and that there is no preemption. The physical infrastructure is composed of a set of heterogeneous physical machines (PM), and each one has a price per time unit. A VM instance is deployed onto a single PM. A VM can become idle in the last time intervals if the period necessary to execute the task, or its deadline, is smaller the billing period. If there are free resources, there is no delay between the time that a new VM instance is requested and when it becomes available to execute the tasks.

B. Application Model

This paper focuses on scientific workflows that can be modeled as DAGs, a directed graph with no cycles where the nodes in the graph represent the computational tasks and the edges represent the temporal relations between the tasks. A DAG can be modeled by a tuple $G =< T, E >$, where T is the set of n tasks of the workflow, such that $T = \{t_1, t_2, ..., t_n\}$. The set of edges E represent the data dependencies among tasks, where each dependency indicates that a child task cannot be executed before all its parent tasks finish successfully and transfer the required child input data. It is assumed that all workflow data is stored in a shared cloud storage system, such as Amazon S3, and that intermediate data transfer times are included in task runtimes. It is also assumed that data transfer times between the shared storage and the VMs are equal for different VMs instances so that task placement decisions do not impact the

runtime of the tasks. The runtime estimates and the CPU computational needs for the workflow tasks are known. Due to heterogeneity of PMs, each task may have a different execution time on each processor. Only workflows for which all tasks are finished by the deadline are considered to be complete.

C. Scheduling Model

The scheduling system model depends on the resource model, application model, and performance criterion for scheduling. As represented in Figure 1, this paper considers the typical scenario where users submit sets of workflow applications at different instants of time, each of which has specific individual time and cost constraints. The objective is to find a schedule map for each workflow application that meets its user-defined QoS constraints (i.e., the completion time of the workflow shall not exceed the deadline constraint, and the total cost must not be higher than the available budget). The time the application takes to complete, i.e. execution time or makespan, includes the waiting time, the execution time, and data transfer time of a given workflow. So that the scheduler can find feasible solutions, the deadline and budget must be properly negotiated between the user and the provider in a range of feasibility values.

The scheduler may consider the reutilization of idle partial time intervals (i.e., idle VM instances), in order to decrease the workflow execution cost.

IV. Evaluation Scenario

A. Experimental Workflows

The Pegasus project made available a set of realistic workflows from diverse scientific applications. These workflows are available in Directed Acyclic Graph in XML (DAX) format, under different sizes (i.e., number of tasks). A DAX file characterizes in detail the structure, data and computational requirements for a specific workflow. In particular, we chose the Montage workflow from the Pegasus project, a general engine for computing mosaics of input images in the Flexible Image Transport System (FITS) format. Being a realistic workflow, the Montage application have been used in current research [12], [13].

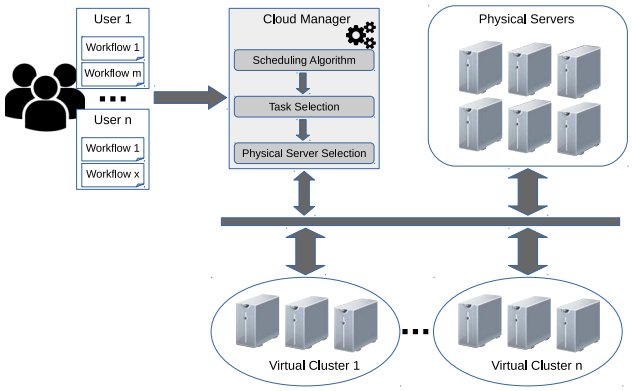

Figure 1: Distributed system architecture.

B. Algorithms

The MW-DBS algorithm works in three steps: (i) fill a pool of ready-to-execute tasks; (ii) task selection; and (iii) processor selection. In the first step, a priority is assigned to each task, based on its critical path. From each workflow application, a single ready task with the highest priority is selected and added into the pool. To determine which task should be selected for scheduling among all ready-to-execute tasks, the second step assigns a priority to each task, which is inverse to its deadline and proportional to the ratio of the number of scheduled tasks to the total number of tasks in the workflow application. The task with the highest priority is selected for scheduling. The third step selects a processor to run the task, based on the the combination of the two QoS factors, time and cost, in order to obtain the best balance between time and cost minimum values.

Unlike MW-DBS, which assumes that the cost imputed to the workflow execution is only the effective used time, four other variants were developed. These four scheduling alternatives follow a more realistic approach, in the way they are aware that a charged time interval may not be completely used if the workflow task finishes executing before the end of the billing period. The scheduling variants are explained below.

1) Reuse User idle time Slots MW-DBS (RUS-MW-DBS): It schedules tasks to user owned idle VM instances, in order to exploit user owned idle last time interval fractions that have been charged. In case there are no user owned idle VM instances available at the scheduling time, a new VM instance is created to run the task, which remains available during the Cloud billing cycle. Additionally, the scheduling algorithm eventually prolongs the life of the VM instance that will execute the task, by guaranteeing multiple time intervals on the same PM resources, in order to accommodate the entire task execution.

2) Reuse Others idle time Slots MW-DBS (ROS-MW-DBS): It extends the scheduling policy in RUS-MW-DBS, in the sense it can schedule a task to an idle last time interval owned by any user. If a first user task is scheduled to a second user owned VM instance, then the first user is charged according to the period of time necessary to execute the task.

3) Unlimited idle time Slots for Free MW-DBS (USF-MW-DBS): This scheduling algorithm alternative equals to ROS-MW-DBS, except that idle VM instances are available at no charge, no matter the user they belong to.

4) Dynamic MW-DBS (DYN-MW-DBS): It applies exact charging, by creating VM instances as needed, and terminating them at the end of task execution. In this regarding,

it is assumed that billing periods equal the tasks finishing times.

C. Simulator

To evaluate the advantages for cost-efficiency schedules in exploiting unused last time intervals, we used the discrete-event cloud simulator introduced in [14], which implements the distributed architecture of Figure 1. Discrete-event simulation allows us to ensure the repeatability and reproducibility of large-scale experiments, for a wide range of application configurations in a reasonable amount of time.

The simulator consists of two main entities: the cloud manager and the Scheduler. The Cloud manager starts and terminates clusters of VMs to serve users' requests. It is also its duty to manage the execution of individual tasks. The Scheduler is responsible for the scheduling of tasks and the provisioning of VMs based on the applied algorithm. The simulator reads workflow description files in DAX format, from the Pegasus project [15]. The VMs in the simulator have a single core and execute tasks sequentially.

D. Experimental Parameters

Aiming at observing the characteristics of the proposed methodology for the different scheduling algorithms, we define a value for deadline and budget constraint parameters for each individual workflow application. The deadline parameter is defined by Eq. 1.

$$Deadline = min_{time} + \alpha_D \times (max_{time} - min_{time}) \quad (1)$$

where min_{time} and max_{time} correspond to the lowest and highest execution time, respectively. They are determined based on the lowest and highest possible makespans for an infinite number of CPUs, in which is considered the processing time for the critical path. In this paper, we assume that the average communication time between tasks and their critical parents is practically zero.

The budget parameter is defined by Eq. 2.

$$Budget = min_{cost} + \alpha_B \times (max_{cost} - min_{cost}) \quad (2)$$

where min_{cost} and max_{cost} represent the absolute highest and lowest possible costs for executing the application, respectively. These two parameters are calculated by summing the maximum and the minimum execution costs for each task, respectively.

To proceed with the simulations, a set of 20 users was created, each of which had a list of a randomly number (between 1 and 10) of workflow applications to execute, totaling 104 workflows. Each workflow structure consisted of Montage application with 25 tasks, but the tasks execution times were multiplied by 10 because they are quite small. The deadline and budget constraints to each workflow were determined randomly by varying $\alpha_D \in \{1, 1.5, 2\}$, and $\alpha_B \in \{0.15, 0.5, 1\}$. The idea is to cover a broad parameter space, from tight constraints, where only a small number of workflows can be completed, to more liberal constraints where almost all of the workflows can be completed. The list of workflows was submitted to the simulator. Users arrive to the system at random time instants, ranging between 40% and 120% of the minimum execution time of the Montage application. Once users arrive to the system, they start submitting workflows at random time instants that range from 10%, 30% and 50% of completed tasks (i.e., a new workflow is inserted by its user when the corresponding percentage of tasks from its last workflow currently in the system is completed or the last one was failed).

Table I: Characterization of the Cloud infrastructure (granularity of time intervals is 1 hour).

PM Class	Quantity	GFlops	Cost ($)	Full Power (Watt)
A	13	4.040	0.055	264
B	13	8.080	0.110	273
C	12	12.12	0.165	289
D	12	16.16	0.220	302

The underlying physical infrastructure to process the tasks is described in Table I. Basically, it is composed of 50 PMs, grouped in 4 classes in which PMs are homogeneous (PMs are heterogeneous across different classes). The processing power in terms of GFlops is presented in column "GFlops". Billing is by the hour with partly used hours incurring a full hour charge. The price for VM instance per billing period is defined in column "Cost"[1]. Relating columns "GFlops" and "Cost" together, we observe that VM instances running on the most powerful PMs are costly, whilst the VM instances scheduled on less powerful PMs are cheaper. The power consumed by a PM, working at full CPU capacity, is defined in column "Full power", and is based on real data provided by the results of the SPECpower benchmark[2]. We assume that the power consumed by a PM evolves linearly with CPU utilization, decreasing to 70% of its maximum power in idle state [14]. The energy cost (per kWh) was defined to $0.189, which consists in a typical electricity price. A set of simulations is conducted by varying the time interval granularity in $\{1, 5, 10, 30, 60\}$ minutes.

E. Performance Metrics

In order to analyse the proposed methodology, it is necessary to introduce the metrics that can be used to score the performance of the different algorithms. In this regarding, we defined three metrics, as described bellow.

1) Number of Workflows Successfully Executed (E_W): A workflow is considered successfully executed if all its tasks are finished within the budget and before the deadline.

[1]Google Compute Engine. https://cloud.google.com/compute/pricing
[2]The SPECpower benchmark. http://www.spec.org/power_ssj2008/

2) Total Cost to Users (E_C): Expressed in $, is the sum of all billing periods for all users. Costs related to reused partial last time intervals agree with the scheduling algorithm policies (for example, USF-MW-DBS implies zero monetary costs on idle VM instances reutilization).

3) Energy Consumption (E_E): Represented in KWh, it expresses the cost to resource providers to execute the workflows, and contributes to operational costs. The energy consumption is determined by multiplying the power consumption of the computing infrastructure, in kilowatt, by the number of hours considered.

V. Results and Discussion

Figure 2 shows the results for the number of workflows successfully executed, while varying the billing period in $\{1, 5, 10, 30, 60\}$. As we can observe, all the scheduling algorithms perform similarly for small billing periods, by finishing all the submitted workflow applications. However, as the billing period increases beyond 10 minutes, RUS-MW-DBS is unable to successfully execute all the submitted workflows. In fact, we noticed that RUS-MW-DBS is the algorithm that degrades more as: (i) the billing period increases; and (ii) the rate of arriving workflows increases. Both factors contribute to exhaust the available resources. Contrary to the idea that Cloud providers offer virtually unlimited amount of resources, the number of PMs is limited. Since RUS-MW-DBS is unable to use other users' idle VMs, long billing periods exacerbate scarcity of resources, because other users' VM instances will remain idle for longer. In turn, ROS-MW-DBS strategy, which implies to pay the corresponding fraction of time intervals used if they are owned by a different user, is able to accomplish with all users' requests, even for longer billing periods.

Figure 3 presents the cost to end users to use the infrastructure to execute their workflows applications. Excepting DYN-MW-DBS algorithm, the graph shows that users pay

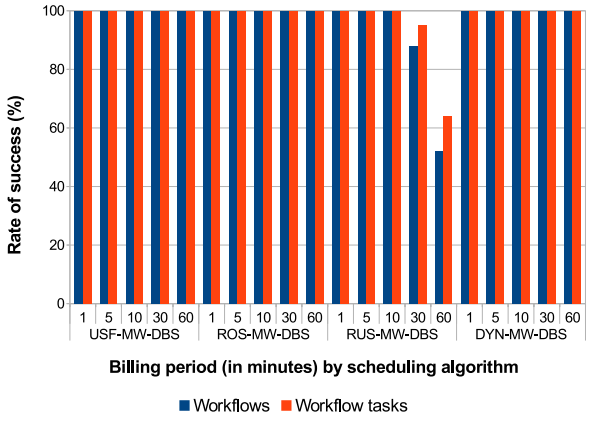

Figure 2: Number of workflow applications and workflow tasks successfully executed.

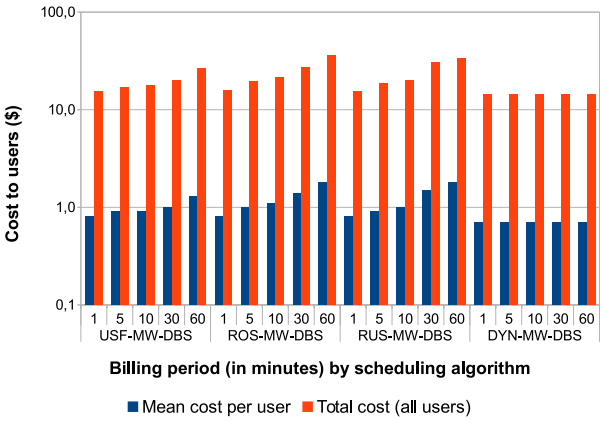

Figure 3: Total cost to users.

more as the billing period increases. In fact, a user will be charged for the entire billing cycle even if it is not fully utilized, and bigger billing cycles correspond to bigger prices. What is more, as the billing period increases beyond 10 minutes, ROS-MW-DBS gets less costly in average to end users than RUS-MW-DBS. Considering the total cost for a billing cyle of 60 minutes, RUS-MW-DBS is less costly than ROS-MW-DBS to end users, because the former is unable to lease time intervals due to scarcity of resources, with corresponding impact in the rate of successfully completed workflows (see Figure 2). In the case of DYN-MW-DBS, the cost remains practically constant despite the billing cycle variation, because DYN-MW-DBS is charged according to the tasks' finish time. This strategy implies significant less profit to provider. In turn, USF-MW-DBS presents the best results comparing to ROS-MW-DBS and RUS-MW-DBS, since it schedules tasks to idle VM instances at zero cost.

Figure 4 shows the energy consumption (KWh) and the electricity bill ($) when executing the users' workflow applications. These two factors heavily contribute to increase the operational costs to the provider and carbon footprints to the environment. The graph shows that the best results are obtained for DYN-MW-DBS. Both USF-MW-DBS and ROS-MW-DBS obtain similar results, reaching a maximum discrepancy of 1.24% in energy consumption for the particular case of a billing cycle of 30 minutes. Comparing to RUS-MW-DBS, ROS-MW-DBS consumes 21.60%, and 13.87% less energy, for billing cycles of 30 and 60 minutes, respectively. ROS-MW-DBS cannot compete with DYN-MW-DBS, nor with USF-MW-DBS. However, while the former implies creating and terminating VMs on-the-fly, and the overhead can be a concern due to boot time and consequent delay in the execution of tasks, the last decreases the providers profit and stimulates inequity among users. As a result, ROS-MW-DBS benefits both users and providers, in the sense it is able to accomplish with users requests with reasonably cost, at lower expenses for Cloud providers.

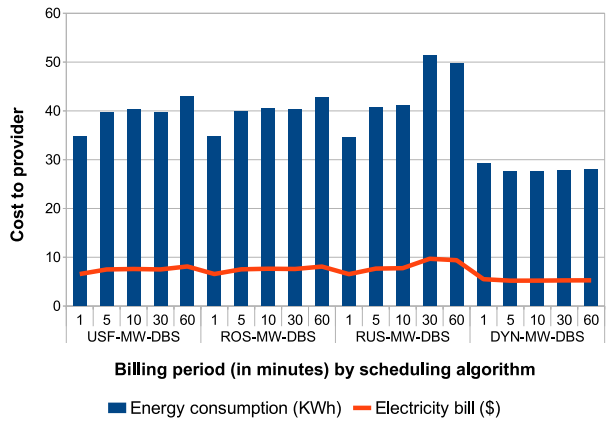

Figure 4: Energy consumption and corresponding electricity bill.

VI. Conclusions

In this paper we proposed to reuse the idle fractions of last time intervals in current Cloud providers. This is a existing problem in nowadays providers due to the common hourly charging granularity. Such billing period turns out to be long for many applications, causing leased resources to go idle.

We have conducted a set of experiments using the Montage workflow, a realistic scientific application, and a set of scheduling algorithms that apply different policies regarding the use of idle VM instances. The experiments showed that providers can become 21.60% more competitive in terms of operational costs, if they assume a policy of reusing already charged fractions of last time intervals. Such fractions can be accessed by a user that is different from its owner, which in turn pays according to the fraction of time effectively used.

This paper consists in a preliminary work on discovering the advantages from reusing fractions of charged last time intervals. As future work, we intend to continue seeking for viable policy and pricing model alternatives, properly supported by efficient scheduling algorithms, that are able to decrease the final resource leasing price to end users, and of improving the competitiveness of providers.

References

[1] 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, no. 5, pp. 528–540, 2009.

[2] M. Wieczorek, A. Hoheisel, and R. Prodan, "Towards a general model of the multi-criteria workflow scheduling on the grid," *Future Generation Computer Systems*, vol. 25, no. 3, pp. 237–256, 2009.

[3] H. Arabnejad and J. G. Barbosa, "List scheduling algorithm for heterogeneous systems by an optimistic cost table," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 25, no. 3, pp. 682–694, 2014.

[4] M. R. Garey and D. S. Johnson, "Computers and intractability: a guide to the theory of np-completeness. 1979," *San Francisco, LA: Freeman*, 1979.

[5] E. N. Alkhanak, S. P. Lee, and S. U. R. Khan, "Cost-aware challenges for workflow scheduling approaches in cloud computing environments: Taxonomy and opportunities," *Future Generation Computer Systems*, 2015.

[6] H. Arabnejad and J. Barbosa, "Multi-workflow qos-constrained scheduling for utility computing," in *Proceedings of the 2015 IEEE 18th International Conference on Computational Science and Engineering*. IEEE Press, 2015, pp. 137–144.

[7] S. Niu, J. Zhai, X. Ma, X. Tang, and W. Chen, "Cost-effective cloud hpc resource provisioning by building semi-elastic virtual clusters," in *High Performance Computing, Networking, Storage and Analysis (SC), 2013 International Conference for*. IEEE, 2013, pp. 1–12.

[8] S. Yi, D. Kondo, and A. Andrzejak, "Reducing costs of spot instances via checkpointing in the amazon elastic compute cloud," in *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*. IEEE, 2010, pp. 236–243.

[9] A. Marathe, R. Harris, D. Lowenthal, B. R. de Supinski, B. Rountree, and M. Schulz, "Exploiting redundancy for cost-effective, time-constrained execution of hpc applications on amazon ec2," in *Proceedings of the 23rd international symposium on High-performance parallel and distributed computing*. ACM, 2014, pp. 279–290.

[10] B. Javadi, R. K. Thulasiram, and R. Buyya, "Statistical modeling of spot instance prices in public cloud environments," in *Utility and Cloud Computing (UCC), 2011 Fourth IEEE International Conference on*. IEEE, 2011, pp. 219–228.

[11] D. Ma and J. Huang, "The pricing model of cloud computing services," in *Proceedings of the 14th Annual International Conference on Electronic Commerce*. ACM, 2012, pp. 263–269.

[12] S. Abrishami, M. Naghibzadeh, and D. H. Epema, "Deadline-constrained workflow scheduling algorithms for infrastructure as a service clouds," *Future Generation Computer Systems*, vol. 29, no. 1, pp. 158–169, 2013.

[13] E. Deelman, G. Singh, M. Livny, B. Berriman, and J. Good, "The cost of doing science on the cloud: the montage example," in *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*. IEEE Press, 2008, p. 50.

[14] A. M. Sampaio and J. G. Barbosa, "Towards high-available and energy-efficient virtual computing environments in the cloud," *Future Generation Computer Systems*, vol. 40, pp. 30–43, 2014.

[15] E. Deelman, G. Singh, M.-H. Su, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, G. B. Berriman, J. Good *et al.*, "Pegasus: A framework for mapping complex scientific workflows onto distributed systems," *Scientific Programming*, vol. 13, no. 3, pp. 219–237, 2005.