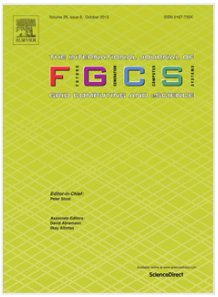
Accepted Manuscript



Taxonomies of workflow scheduling problem and techniques in the cloud

Sucha Smanchat, Kanchana Viriyapant

PII: S0167-739X(15)00177-6
DOI: http://dx.doi.org/10.1016/j.future.2015.04.019
Reference: FUTURE 2763

To appear in: *Future Generation Computer Systems*

Received date: 12 January 2015
Revised date: 25 March 2015
Accepted date: 30 April 2015

Please cite this article as: S. Smanchat, K. Viriyapant, Taxonomies of workflow scheduling problem and techniques in the cloud, *Future Generation Computer Systems* (2015), http://dx.doi.org/10.1016/j.future.2015.04.019

***Highlights (for review)**

We propose taxonomies of cloud workflow scheduling problem and techniques.

Aspects and classifications unique to cloud workflow scheduling are identified.

Several techniques are reviewed and classified based on the proposed taxonomies.

Some issues of future concern in cloud workflow scheduling are discussed.

Taxonomies of Workflow Scheduling Problem and Techniques in the Cloud

Sucha Smanchat
sucha.smanchat@acm.org, sucha.s@it.kmutnb.ac.th

Kanchana Viriyapant
kanchana.v@it.kmutnb.ac.th

Faculty of Information Technology, King Mongkut's University of Technology North Bangkok
1518 Pracharat 1 Road, Bangsue, Bangkok, Thailand

ABSTRACT

Scientific workflows, like other applications, benefit from the cloud computing, which offers access to virtually unlimited resources provisioned elastically on demand. In order to efficiently execute a workflow in the cloud, scheduling is required to address many new aspects introduced by cloud resource provisioning. In the last few years, many techniques have been proposed to tackle different cloud environments enabled by the flexible nature of the cloud, leading to the techniques of different designs. In this paper, taxonomies of cloud workflow scheduling problem and techniques are proposed based on analytical review. We identify and explain the aspects and classifications unique to workflow scheduling in the cloud environment in three categories, namely, scheduling process, task and resource. Lastly, review of several scheduling techniques are included and classified onto the proposed taxonomies. We hope that our taxonomies serve as a stepping stone for those entering this research area and for further development of scheduling technique.

1. Introduction

Scientific workflows have been employed to streamline computational experiments to be executed automatically so that scientists are relieved of technical impediment. To cater for the computational power required to execute large scale experiments, these workflows have successfully been executed in the grid computing environment, which is a distributed system based on resource sharing and coordination. Such execution is usually planned and scheduled so that the workflow is efficiently executed on grid resources. For this purpose, scheduling techniques for scientific grid workflows have been studied and developed over many years [1, 2].

With the emergence cloud computing, which offers computing resources elastically on demand [3], scientific workflows, like other computer applications, can benefit from virtually unlimited

resources with minimal hardware investment. With such advantages, cloud computing has attracted much attention and the research in workflow scheduling has thus shifted to workflow execution in the cloud environment. Unlike grid resources, which are usually utilized free of charge according to sharing agreements, cloud resources such as virtual machines are associated with costs according to their usage. Also, resources in the cloud are provisioned in different ways. Therefore, workflow execution in the cloud needs a scheduling technique that is different from those used by the grid workflows.

Since the birth of cloud computing, several cloud workflow scheduling techniques have been proposed. These techniques consider many aspects of cloud environment, which can be different due to the flexibility of cloud computing. For example, some techniques assume that a workflow is executed in hybrid cloud environment while others assume only a public cloud platform. These different considerations influence the workflow scheduling problem leading to the techniques of different designs.

This paper aims to identify these aspects of consideration in cloud workflow scheduling in the form of taxonomies. In the past, Wieczorek , Hoheisel, and Prodan [4, 5] presented taxonomies of workflow scheduling problem in the grid computing domain. Although, many aspects and classifications are shared in both the grid and the cloud contexts, new considerations arise in scheduling cloud workflow. Bittencourt, Madeira, and Fonseca [6] identified some issues of concern when scheduling in cloud environment but they focus mainly on resource characteristics and hybrid cloud execution. Alkhanak, Lee, and Khan [7] proposed a classification of challenges in cloud workflow scheduling focusing more on scheduling objectives and functionalities of workflow system architecture.

Other literature surveys of cloud workflow scheduling mostly focus only on the scheduling objective and the nature of each technique, which do not provide a comprehensive view of the research in this field [8-11]. In this paper, we propose taxonomies that complement the taxonomies of grid workflow scheduling in [4] to accommodate new issues introduced by cloud computing. In our review of cloud workflow scheduling literature, several new aspects can be identified and categorized, such as different types of resource usage costs and provisioning.

The structure of this paper is as follows. Section 2 defines a generic definition of cloud workflow scheduling problem. Section 3 defines and explains the aspects and the classifications that are used to form our taxonomies of cloud workflow scheduling problem and techniques. Some notable

techniques are presented in Section 4 along with a comparison summary. The paper is then concluded in Section 5.

2. Cloud Workflow Scheduling Problem

This section presents generic definitions related to cloud workflow scheduling; different techniques may use different forms of these definitions. Generally, a workflow is modeled as a directed acyclic graph (DAG). A workflow is defined as $w = (T, E)$ where T is a set of tasks represented by vertices in the DAG and E is a set of precedent dependencies represented by edges in the DAG. A task dependency is defined as $e = (t_i, t_j)$ where $t_i \in T$ is a predecessor task of $t_j \in T$ and $t_i \neq t_j$. A task t can start its execution if and only if all of its predecessor tasks have completed their execution.

Cloud workflow scheduling is then the mapping $T \rightarrow R$ from each task $t \in T$ to a resource $r \in R$ so that the specified criteria are met. In this domain, resources in the cloud are usually assumed to be virtual machine instances that are instantiated from machine images readily stored in the cloud (i.e. the concerns related to moving the machine images to the cloud is excluded). A task may require a program or software to execute, which are assumed to be preinstalled on the machine images. Therefore, it is also assumed that a task can be executed by any virtual machine instance type (i.e. any machine specification) by instantiating from the corresponding machine image [12].

Thus, a task t can be defined as $t = (id, MI)$, where id is the identity of t, and $mi \in MI$ is a set of virtual machine images that can perform execution for t. A compute resource r can be defined as $r = (mi, mt)$, where $mt \in MT$ is the virtual machine instance type of r among the instance types in MT that are offered by cloud providers.

3. Taxonomies of Cloud Workflow Scheduling Problem and Techniques

This section identifies and explains the aspects and classifications of workflow scheduling in the cloud environment. Some of these aspects were identified in the taxonomies of grid workflow scheduling presented by Wieczorek , Hoheisel, and Prodan [4] and in a review by Bittencourt, Madeira, and Fonseca [6]. Due to the different nature of the grid and of the cloud in resource provisioning, new considerations in the scheduling problem and techniques arise. Also, as an initial work, we have identified a set of information required to implement a cloud workflow scheduler in [13]. To avoid reinventing the wheel, our taxonomies complement and, where possible, augment the taxonomies of grid workflow scheduling in [4].

The aspects presented herein are extracted mainly from research literature in cloud workflow scheduling domain and are separated into three main groups namely: *scheduling process*, *task* and

resource partially following the taxonomies in [4]. Since cloud computing introduces the new compute resource paradigm, the concentration is more on the aspects in the *resource* group than the other two.

3.1. Taxonomy of scheduling process

This group contains aspects related to the scheduling process of scheduling algorithms, which extend the taxonomy in [4]. These aspects are shown in Figure 1. Apart from our classification, Malawski et al. [11] also differentiates whether virtual machine instances are provisioned statically or dynamically during the scheduling process.

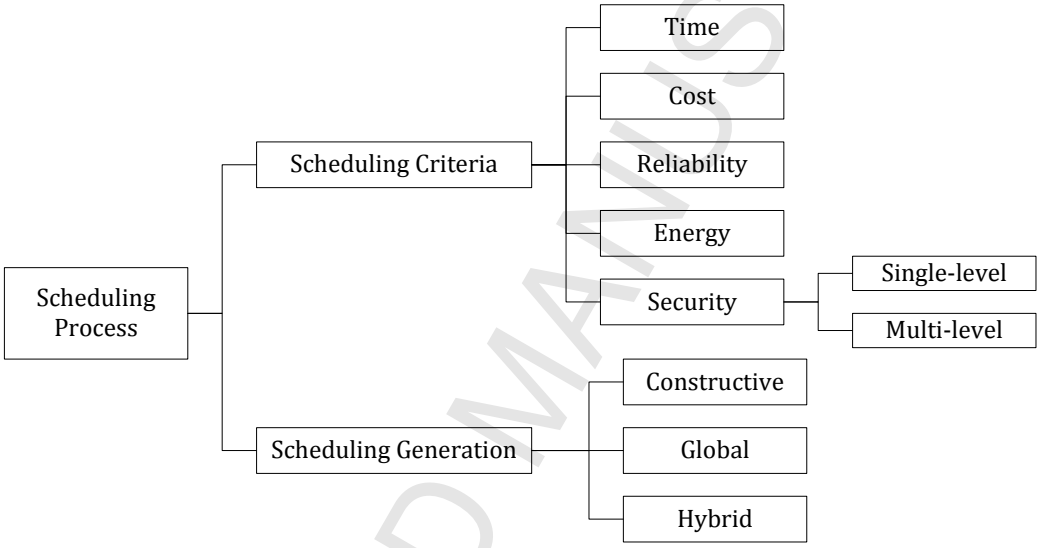


Figure 1 Taxonomy of cloud workflow scheduling process

3.1.1. Scheduling criteria

Scheduling criteria influence the design and approach of scheduling techniques. In contrast to grid workflow scheduling where minimizing makespan (or time) is dominant [1, 14-17], most of the cloud workflow scheduling techniques are multi-objectives, in which time and cost are considered together in the scheduling [6, 18, 19]. Nevertheless, other objectives have also been considered.

Time

Makespan or the total execution time of a workflow is a dominating objective in most scheduling techniques since the age of grid computing. Time objective can be specified as a hard constraint such as deadline and as a soft constraint to be minimized in a best-effort manner.

Cost

Cost has become an important objective in cloud workflow scheduling research. The total cost incurred by workflow execution can comprise many cost components such as compute cost and data transfer cost, which will be explain in subsequent sections. A budget can be set as a hard constraint [18, 20-22]. However, it is more common that the cost is specified as a soft constraint to be minimized while satisfying a fixed deadline [18, 19, 23, 24]. In a relaxed case, both time and cost are not strictly specified [25], thus, it is possible to obtain many solutions with tradeoffs between the two constraints [26]. Cost-aware scheduling techniques have also been introduced for utility grid prior to the advent of cloud computing [24, 27-32]. However, the cost model, cost components and resource provisioning in the cloud are, though not entirely, different.

Reliability

Apart from the most common time and cost criteria, workflow execution reliability is also addressed. This objective ensures that resources selected in a schedule will likely to complete the tasks scheduled to them. Task execution failure is usually handled by restarting and replicating the task. However, both mechanisms may respectively cause the waste of time and compute resources [33-35]. Also, for cloud resources that are provisioned with no guaranteed reliability such as Amazon EC2 Spot Instance [36], this objective could be beneficial to ensure task completion.

Energy Consumption

With the growing environmental concern, minimizing energy consumption or carbon footprint has begun to receive attention [37, 38]. Although, this issue is not unique to cloud computing, it also has attracted researchers in this field. For example, this objective is optimized as one of the multi-objective scheduling in [25]. Luo and Zhou [39] recently addresses this issue by introducing power consumption model. The model is used to estimate the energy consumed by cloud services so that the algorithm can select, among many schedules that satisfy time and cost constraints, the schedule with minimum energy requirement. Nevertheless, the authors acknowledge that the power optimization is still not applicable in virtual machine abstraction level.

Due to the virtualization and abstraction of physical resources and that this objective is more related to cloud providers [25], workflow schedulers might not be able to directly address this issue unless the information regarding the run-time energy consumption of each virtual machine instance is made available by cloud providers to workflow schedulers [39].

Security

Data security, privacy and governance have become an important issue when an organization decides to adopt cloud computing solution. Although, security mechanisms can be implemented on virtual machine instances and networks, the governance of data may specify regulations or may be required by laws to prevent data from leaving on-premise infrastructures. To address these concerns, security and privacy requirement of data that needs to be processed by a workflow should be defined [20, 40, 41]. This requirement usually imposes a hard constraint on cloud resource selection.

The consideration of security and privacy may be of workflow level and of task level. Assuming the workflow level, the whole workflow can be executed only on the private resources. This assumption is thus out of our scope because it poses a limit on and does not affect the scheduling decision. On the other hand, assuming the task level, each task can be scheduled to a private resource or a public cloud resource according to its security and privacy requirement [42]. We differentiate the security constraint into two subgroups, namely, *single-level* and *multi-level*.

The *single-level* constraint simply specifies whether a dataset (and the task that executes on that particular dataset) requires security. This is assumed by the "SABA" algorithm [20] that differentiates between "movable datasets" and "immovable datasets". Most of the other workflow scheduling techniques usually assume the data of the former type and transfer these data as necessary. The data of the latter type, on the other hand, are not allowed to be transferred and duplicated. The so called "immovable tasks" that process the immovable data must be executed on the resources hosted within the same data centers that hold such particular data. Similar concept is also used in the "Tagged-MapReduce" approach [40], which includes a sensitivity tag in the *Map* process to distinguish between sensitive and non-sensitive data so that the scheduler can restrict sensitive data to be executed on private resources only.

On the other hand, the *multi-level* security model assumes that security requirement can be specified in many levels [42, 43]. As opposed to the previous subgroup, workflow scheduling can be more complicated because datasets and tasks requiring a specific security level can still be executed on virtual machine instances provisioned by public clouds whose images are installed with trusted software or services.

With the consideration of security, the definition of task introduced earlier should be extended. However, this depends on how the security is considered in each workflow system. For example, only a list of trusted and secured virtual machine images can be specified for each task. Also, a

security flag can be included in the task definition to restrict execution on private or trusted clouds. With the rapid development and the movements influencing the security in cloud computing, our proposed classification is subject to future refinement.

3.1.2. Schedule generation

This section describes the aspect of how scheduling techniques build a schedule. Generally, a scheduled can be generated in two different ways as previously introduced in [1] as heuristics based and meta-heuristics based techniques in grid workflow scheduling domain. In this paper, we generalize the classification in [1] so that further techniques may fit better in our taxonomy. Note that this aspect is different from the "dynamism" of the scheduling process defined in [4] as their concern is on the time a scheduling process is invoked.

Constructive scheduling

A schedule can be gradually built by iteratively mapping a task to a resource. We refers to this type as *constructive*, which are common in list scheduling techniques [1, 18, 19, 23, 44, 45]. Workflow structure can be taken into account in the scheduling by ranking tasks, determining the critical path and workflow partitioning [19, 23, 45, 46].

Global scheduling

The other type, referred to as *global scheduling*, starts with a complete schedule. The initial schedule can be obtained by different methods, such as randomly selecting a resource for each task, and is further optimized based on the scheduling criteria. The techniques usually involve meta-heuristic algorithms [47] such as genetic algorithm [28, 48], ant colony optimization [49] and particle swarm optimization [50, 51].

Hybrid

In the hybrid approach, sub-global scheduling can be used in constructive scheduling process. By applying a meta-heuristic algorithm on a subset of the tasks in a workflow, the resulting schedule could be more efficient as (a subset of) tasks are scheduled at the same time. The time required to obtain a solution in a sub-global scheduling also decreases because the problem space, as opposed to purely global scheduling, is reduced [25, 52].

3.2. Taxonomy of tasks

Scientific workflows themselves do not significantly change with the evolution of cloud computing. In this section, we merely discuss the implication of cloud computing on the classification of *task-*

resource mapping introduced in [4] as depicted in Figure 2. This aspect refers to the number of resources, or virtual machine instances, that are required by a workflow task.

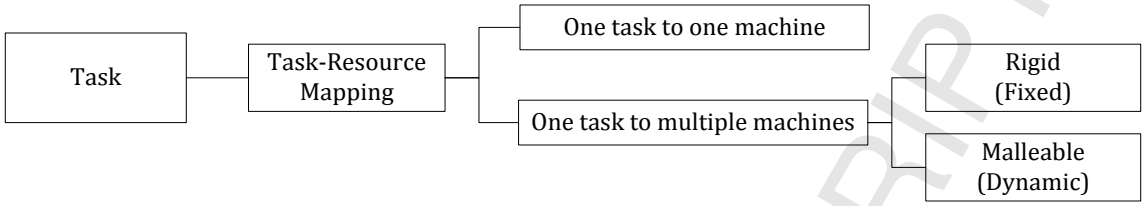


Figure 2 Taxonomy of cloud workflow task

One task to one machine

Previously in the grid workflow scheduling domain, a task is usually assumed to occupy one compute node for execution [27, 53-55] and many of the existing scheduling techniques for cloud workflow still implicitly assume that each workflow task is executed by only one virtual machine instance [6, 23, 46, 52]. This category is also referred to as "*rigid*" in [4].

One task to multiple machines

With MPI and MapReduce programming models [56], it is possible that a workflow task requires multiple nodes or virtual machine instances for its parallel execution [57]. This category can be further divided into "*rigid*", where the number of required instances is fixed during execution, and "*malleable*", where the number can be changed during execution [4, 19]. This consideration is still not common in cloud workflow scheduling literature. Due to the scheduling complexity caused by this assumption, only few algorithms such as the "PBTS" algorithm [19] and its predecessor "BTS" algorithm [45] address this issue. In this case, the definition of task can be refined as $t = (id,\ MI,\ mr)$, where *mr* specifies the number of virtual machine instances required by *t* [19].

With this consideration, a scheduling technique needs to estimate the number of virtual machines required in each time interval [19, 45] in order to determine cost. The scheduling can be more complicated. For example, if tasks with different numbers of required virtual machine instances are to be scheduled in the same interval, a scheduler first needs to determine the number of instances to be instantiated. If all of the required instances are instantiated at once, the cost would be maximal. In order to minimize the cost, the scheduler may choose to instantiate fewer instances, in which case the scheduler needs to properly prioritize the tasks so that, for example, a task requiring more instances is not overly delayed by tasks requiring fewer instances.

Another challenge imposed by this consideration is the management of the running virtual machine instances. Without an accurate estimation of the number of instances required, a scheduler may switch off some instances to save cost while an incoming task may required them for execution. This may result in a higher cost due to the running instances waiting idly for sufficient number of instances to be restarted [57], and possibly a longer overall makespan. This challenge also applies to one-task-to-one-machine mapping but to a lesser extent.

3.3. Taxonomy of resources

Because, in cloud computing, resources are provisioned elastically on-demand with associated costs, many new aspects of resources for workflow scheduling have arisen. These aspects are depicted in Figure 3.

3.3.1. Cloud execution model

The deployment model of a cloud affects how workflows are executed. The nature of private cloud (and community cloud) in general is analogous to that of the grid computing, with the exception of resources being virtualized leading to a higher degree of homogeneity among resources. Because private cloud resources are provisioned internally, the cost of using private cloud is usually assumed to be zero [18, 44]. Therefore, we only consider hybrid cloud and public cloud environment for the workflow execution in this taxonomy.

Execution in hybrid cloud

Hybrid cloud model combines both private cloud and public cloud together for resource provisioning [57]. As explained earlier, using private cloud resources does not incur cost. Thus, the scheduling techniques that assume hybrid cloud context usually start by scheduling all the workflow tasks on private cloud resources to eliminate cost. If such schedule does not meet scheduling criteria (usually deadline), tasks are rescheduled or reassigned to public cloud for a reduction of makespan, while increasing the cost incurred by cloud resource usage [18, 44, 58]. In this environment, the set of resources R during scheduling can also be refined as $R = R_{pri} \cup R_{pub}$, where R_{pri} and R_{pub} are the resources provisioned respectively by private cloud and public cloud [18].

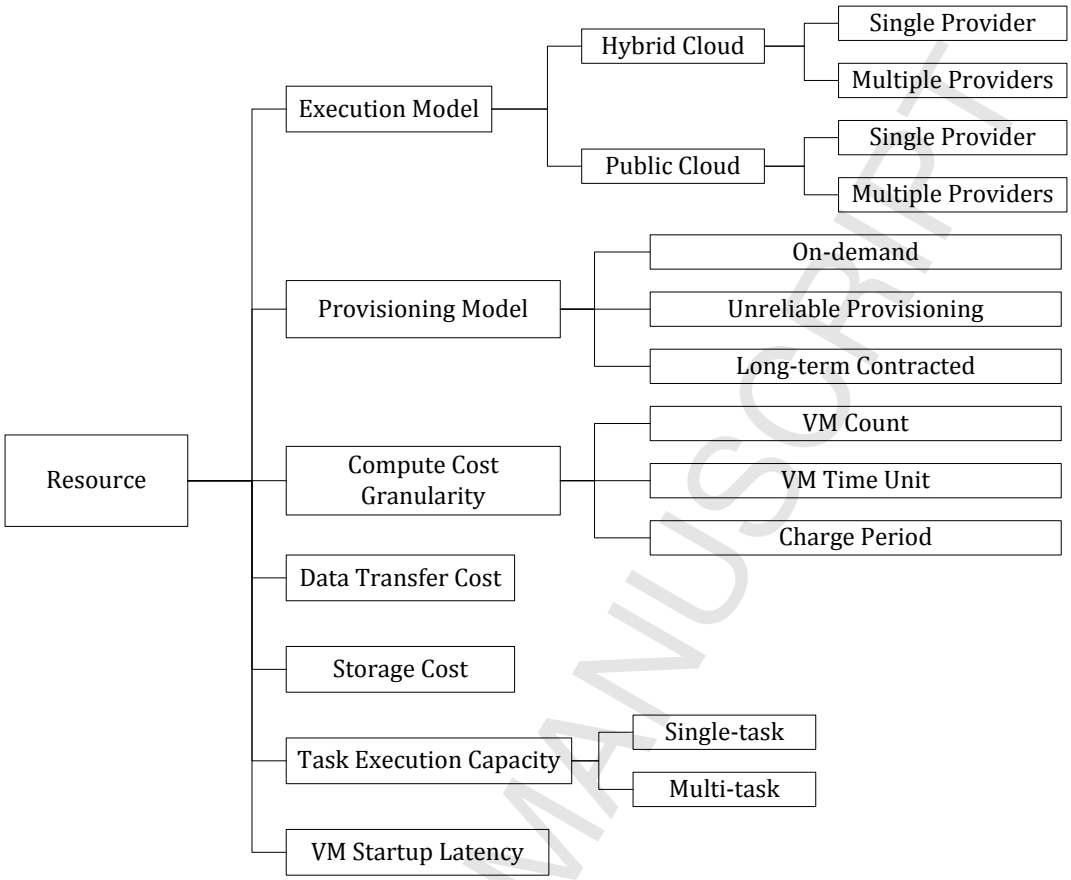


Figure 3 Taxonomy of cloud resource considerations

Execution in public cloud

When executing workflows in public cloud, cost is associated with all resources. The cost of a faster resource is usually higher than that of a slower one (which corresponds to the public cloud services currently available such as Amazon EC2 [59]). Unlike hybrid cloud, there is no zero-cost resource to supplement the execution. Thus, scheduling techniques must cope better with resource selection in order to generate a feasible and efficient schedule. A schedule may be infeasible due to insufficient budget or deadline violation.

In a more complicated scenario, a workflow system may use resources provisioned by multiple cloud providers leading to "intercloud" execution [60, 61]. This may allows for a more cost-efficient schedule by selecting virtual machine types from different providers that best match the computing requirement of workflow tasks. In this environment, the set of resources R provisioned by n

providers can also be refined as $R = \bigcup_{p=1}^{n} R_{pub}^{p}$, where R_{pub}^{p} is a set of resources provisioned by public cloud provider *p*.

However, the cost for transferring data out of cloud data centers stipulated by public cloud providers is usually not negligible. Data-intensive scientific workflows may suffer from increased cost if data are transferred back and forth due to data dependencies between tasks. Thus, the scheduling techniques that assume multiple cloud providers should include data transfer costs in the scheduling process [52].

3.3.2. Provisioning model

This aspect refers to how virtual machine instances are provisioned to users. Many of the cloud workflow scheduling techniques try to exploit different price models offered by Amazon EC2 [59] including standard on-demand instance, Spot Instance [36], and reserved instance. These three models provision virtual machine instances in different ways. Thus, our classification of cloud resource provisioning models is based on Amazon EC2.

On-demand resource

The on-demand provisioning is the standard model that is considered by most existing scheduling techniques. Unlike grid resources, which are assumed to be limited during workflow execution, virtual machine instances can be launched and terminated at any time. Because of this, scheduling algorithms need to estimate an optimal number of instances to be allocated before an execution [62] (if not specified by users) and to determine if additional instances are required during the execution [12, 63]. Those instances that no longer contribute to the workflow execution should also be switched off to save cost [11, 64]. Of the same instance type, this model, which guarantees reliability, incurs the highest cost per hour among the three provisioning models.

Unreliable resource

Amazon EC2 has offered cheaper virtual machine instances called Spot Instances [36] for which users have to bid. The reliability of this type of resource is not guaranteed because a running virtual machine instance may be outbid at any time and the computation performed on the outbid instance may be lost. However, according to the bid price analysis in [65], it is possible to maintain high availability with sufficiently high bid. Recent techniques have thus begun to consider the use of such resources to further reduce compute cost of executing workflows. For example, the "DCP-C" algorithm [65] makes effective use of Spot Instances [36] to achieve substantial cost reduction of workflow execution.

The inclusion of this type of resources reintroduces the challenge analogous to the grid environment that a resource may become unavailable during workflow execution. Thus, a scheduler needs to consider fault tolerant or reliability factor [34] in the scheduling process and may employ a prediction mechanism to decide if cheap unreliable resources should be used at particular times during execution [65, 66].

Long-term contracted resource

Amazon EC2 offers reserved virtual machine instances that consumers have to pay for long-term contracts such as one or three years. The inclusion of this type of resources in workflow scheduling can be complicated because the consideration of the compute cost is unclear. Because, in addition to a lower hourly cost, reserved instances are paid upfront for the long-term contracts, it is not rational to allocate a new reserved instance without considering its long-term usage. On the other hand, considering a reserved instance already allocated, it may not be rational to include the (partial) upfront cost of the instance because it may have already been covered by another workflow execution (or another application). So far, only the work proposed by Genez, Bittencourt and Madeira [46] explicitly considers this type of resource, but does not clearly address the cost concern.

3.3.3. Compute cost granularity

Because of the nature of cloud services, cost is usually, if not always, considered in most scheduling techniques. Different types of cloud services charge users based on different metrics such as virtual machine usage hours and storage size. In this section, we discuss the cost that is associated with the cloud infrastructure-as-a-service resources.

The execution of workflow tasks on virtual machine instances in the cloud usually incurs compute costs stipulated by cloud providers. The compute cost associated with the usage of infrastructure-as-a-service is usually charged based on an interval. For example, Amazon EC2 instances, as of the year 2014, are charged at hourly rate [59].

However, the assumption of compute cost in existing cloud workflow scheduling techniques varies. This is probably caused by the evolution of cloud services and cloud providers over the years as the assumption of the more recent techniques is usually more complex [19, 23]. We categorized the assumption of compute cost into three granularities.

Virtual machine count

The most primitive assumption of compute cost is in the form of the number of virtual machine instances used for the entire execution. We refer to this type of assumption as *virtual machine count*. This assumption does not contribute to an accurate cost calculation and may not produce a cost-effective schedule because a virtual machine instance may be idle during execution [19, 45]. The examples of scheduling techniques that assume this cost granularity are the RC^2 algorithm [44] and the BTS algorithm [45].

Virtual machine time unit

The compute cost is also expressed in time units of usage, such as virtual machine hour. Usually, the techniques in this group implicitly assume that the execution of a task takes longer than one time unit. The cost incurred by a virtual machine instance is then the product of the time units used and the price per time unit of that virtual machine instance type. Although this assumption is common in this research area, it is still a coarser-grained assumption to simplify scheduling process because the time unit assumed may not correspond to the actual charged period stipulated by public cloud providers. It is possible that the charged period is longer than the actual usage of a virtual machine instance [19]. Thus the unused period incurs cost without virtual machine utilization, resulting in an unnecessarily high cost [23].

Charge period

In more complicated scheduling techniques such as the "PBTS" [19] and "IC-PCPD2" [23] algorithms, the cost consideration is based on the charge period of the cloud provider. With this assumption, scheduling algorithms try to utilize all the leftover of the charge periods and decide whether to terminate machine instances at the end of their charge periods in order to minimize cost [11]. This fine-grained assumption, although allowing for most cost-effective schedules, introduces a complex scheduling process to fit tasks within charged period. Note that to be qualified in this group, a scheduling technique must explicitly use charge period in its scheduling process (i.e. a mere calculation of the final cost based on charge period is insufficient).

However, as cloud computing evolves, the charged period has become shorter from the hour-scale [59] to the minute-scale [67, 68]. For example, as of the year 2014, users are charged for Google Compute Engine by minute with a minimum of 10 minutes [67], while Microsoft Azure does not pose a minimum duration [68]. With the reduction of charge period, such fine-grained assumption,

which increases the complexity of the scheduling techniques, may not be essential in the near future.

3.3.4. Data transfer cost

In addition to the cost incurred by virtual machine usage, there are also costs for transferring data across the boundary of cloud providers. Usually, cloud providers charge negligible or no cost for inbound data coming into their data centers, but define a cost scheme for outbound data [59, 67, 68]. The outbound data cost is charged regardless of the cloud services from which the data originate (although the prices may differ). This corresponds to the most common usage pattern of the Internet where download streams usually outweigh upload streams. For example, developers do not upload data (e.g. system deployment and content updates) as often as web customers accessing or downloading content from the websites.

Additionally, the cost for transferring data within a data center (corresponding to an "availability zone" in some providers such as Amazon [69]) is usually zero and the cost for transferring data across a cloud provider's regions is usually negligible when compared to the cost of outbound data leaving a cloud provider's boundary [59, 67].

Therefore, the only transfer cost that needs to be addressed is the cost for transferring data out of a cloud provider; the other costs are usually assumed to be negligible by most scheduling techniques [18, 52, 70]. However, this consideration is associated with the execution environment explained in Section 3.3.1. If an execution of a workflow takes places within a single public cloud provider [23, 66], the transfer cost would be incurred only by retrieving the final output of the workflow from a cloud data center, which may also be negligible. If a workflow is executed in a hybrid cloud environment, it is possible that data are transferred from a public cloud to local resources several times due to tasks in the same path being scheduled to both cloud resources and local resources. As mentioned earlier, the issue is even more complicated when executing a workflow using resources from multiple cloud providers. Nevertheless, some scheduling techniques still ignore this cost as they assume that the compute cost is significantly higher [19].

3.3.5. Storage cost

Cloud storage cost is usually excluded from the scope of workflow scheduling because the price models defined by commercial clouds, such as Amazon S3 [71] (which is a stepwise model based on the average storage size throughout a month) make the cost calculation difficult. In addition, the

data files in the storage can be shared by many workflow executions making it hard to actually associate the cost to each workflow execution.

The calculation of storage cost is also dependent on how a workflow is executed. Data may be transferred between virtual machine instances directly. In this case, the storage cost is nullified because the block volume storage is already charged together with the virtual machine instance it is attached to. On the other hand, data may first be written to a persistent storage for subsequent sharing [19], in which case the storage cost can be charged.

Nevertheless, some techniques have included the storage cost in the calculation of workflow execution cost. In these work, the storage cost is incurred for storing input, output and temporary data of tasks in a workflow. The cost, however, is calculated from the product of the data size and the price per time unit during the task execution period, which may not be well consistent with the current commercial cloud storage pricing [25, 42, 43].

3.3.6. Task execution capacity

This aspect refers to whether a virtual machine instance can execute more than one task simultaneously. This corresponds to the "task execution" aspect in the taxonomy of grid resource in [4]. Thus, we merely discussed how this aspect is addressed by cloud workflow scheduling techniques.

Single-task

Traditionally, a resource or a virtual machine instance is assumed to execute only one task at any given time. This assumption is common in most workflow scheduling techniques in both the grid and the cloud environments to avoid complexity in the scheduling. However, as pointed out by recent literature, this assumption may lead to an underutilization of resources [72].

Multi-task

As computer hardware advances with multi-core technology, the "multiprogrammed" task execution in [4] is re-introduced by Bittencourt, Madeira, and Fonseca in the "HCOC" algorithm [6, 18] as "multi-core awareness" of scheduling algorithm. This assumption allows for multiple tasks to be executed simultaneously on a single compute node or virtual machine instance. By considering the cores of a compute resource in the scheduling process, it is possible to increase the core utilization to maximize resource usage efficiency, for example, by mapping additional tasks to the virtual machine instance with unused cores [73].

Although scheduling at core granularity can help with cost efficiency, it requires that a scheduling algorithm knows the exact number of cores required by each task in a workflow (and also the number of cores available on each resource). For example, the definition of task can be refined as $t = (id,\ MI,\ cores)$, where *cores* specifies the number of cores required by *t*. A workflow system must be able to provide this information to the scheduler, for example, through user's specification to enable such techniques. The workflow engine must also be able to acquire (and restrict) allocations on virtual machine instances at the core granularity.

Another approach is proposed by Kang et al. [72], which takes a schedule generated by any existing algorithm as input to "virtual machine packing" and "Multi Requests to Single Resource" (MRSR) processes. The two processes group and merge tasks that are scheduled to the same machine instance types at the overlapping time to be executed simultaneously on the same machine instance subject to a workflow deadline. The number of simultaneous tasks, however, is decided only by deadline estimation and a cost reduction constraint.

3.3.7. Virtual machine startup latency

When launching a new virtual machine instance, it takes time to boot up the operating system before being able to perform the scheduled task [11]. This overhead may be insignificant for a long running instance. However, it may become an issue when instances need to be unnecessarily re-launched. A scheduling algorithm (or a scheduler) may terminate a running instance to save cost as it is no longer necessary. But due to subsequent circumstance (e.g. to meeting deadline), the instance may need to be re-launched to accommodate arriving tasks. Also, for small tasks, the overhead for launching new instances may not be justified. These situations magnify the effect of the virtual machine startup latency.

Some techniques have begun to address this latency by including the instance boot up time in the estimation of execution time for cloud resources and by fitting tasks into running instances to avoid launching new ones [65, 66]. Also, Mateescu, Gentzsch and Ribbens [57] introduces an algorithm to determine whether it is cost-effective to terminate a virtual machine instance.

4. A Survey of Workflow Scheduling Techniques in the Cloud

In response to the evolution of cloud computing over the past several years, scheduling techniques have also become more advanced and complicated to address issues identified in our taxonomies. In this section, we present a survey of notable workflow scheduling techniques in the cloud.

4.1. CTC algorithm

In 2009, Liu [74] proposed the "Compromised-Time-Cost" (CTC) algorithm, which contains two variations to scheduling workflow in hybrid cloud. Given a deadline, the "CTC-MC" variation aims to reduce execution cost while meeting the deadline, while the goal of the "CTC-MT" variation is reducing makespan under a given budget. As an early technique, only execution time, data transfer time, and compute cost are considered in the scheduling process to allocate cheaper cloud resources to workflow tasks if the time constraint is not violated. As one of the earliest scheduling algorithm for cloud workflow, the compute cost in this work is defined as a price per task execution, which is not consistent with the current commercial cloud pricing models. The highlight for this technique is that users are notified of the current estimated makespan and cost during workflow execution so that run-time adjustment can be made when needed.

4.2. A particle swarm optimization approach

A particle swarm optimization (PSO) technique is used by Pandey et al. [52]. As a batch-mode algorithm, PSO generates a schedule only for tasks that are ready to execute in each scheduling round. Because PSO is not applied on the whole schedule, the scheduling overhead of meta-heuristics known to be significantly longer than those with simple heuristics [1] can be avoided. The compute cost and the data communication cost are input to the PSO to minimize the overall cost. The technique tries to avoid large data transfers, and is able to balance the load on all resources based on their cost. The algorithm considers the execution time only as the inverse proportion to the compute cost (i.e. a more costly resource can execute a task with less time) thus the time constraint is not explicitly addressed.

4.3. RC² algorithm

Lee and Zomaya [44] proposed an algorithm called "RC²" for (re)scheduling independent tasks to achieve reliable task completion in hybrid cloud thereby reducing completion delay. An initial schedule is first generated based on only private resources such as a private cloud or a grid to avoid public cloud resource usage in order to save cost. During execution, if a task completes later than expected, a delay is identified at that particular private resource that the task is executed on. If the delay causes the global makespan to increase, subsequent tasks that have also been scheduled to that particular resource are rescheduled to cloud resources to compensate the delay. The global makespan is thus analogous to an implicit deadline. Although this algorithm does not directly address task dependency in the form of workflow, it is still applicable as a batch-mode scheduling technique.

4.4. HCOC algorithm

The "HCOC" algorithm was proposed by Bittencourt and Madeira [18] to minimize cost of cloud usage within a deadline. The execution model assumed consists of a private cloud of heterogeneous resources and public clouds. Similar to the RC^2 algorithm, HCOC starts by generating an initial schedule based on only private resources by using an existing grid workflow scheduling algorithm called "PCH" [75], which groups tasks in the same workflow paths and schedules the tasks in each group to the same resource. Rescheduling is triggered when a deadline is expected to be violated to reassign tasks to public cloud resources.

This algorithm is among the early algorithms to consider the multi-core aspect of compute resources to achieve cost efficiency. When selecting cloud resources for the tasks to be rescheduled, the algorithm determines the number of cores required by these tasks and chooses the instance type with the least cost per core. Although the authors mention an overhead while virtual machines wait for data staging in their later work [6], it is not included in this technique.

4.5. DCP-C algorithm

To achieve further cost reduction, Ostermann and Prodan [65] proposed the "DCP-C" algorithm that extended the "Dynamic Critical Path" (DCP) algorithm [76] to schedule workflow onto Amazon EC2 Spot Instances [36] to complement grid resources when necessary. Although Spot Instances do not guarantee reliability, by analyzing the bid price from historical data, the algorithm can maintain a sufficiently high availability with the cost still lower than that of the standard instances. The algorithm selects a resource for each task based on the performance of the instances, the cost of the instances and the available budget, and the reliability of the Spot Instances. However, the number of instances is determined largely by a user-defined resource utilization level. Being used for computation-intensive workflows, this algorithm ignores the data transfer cost but includes the virtual machine overhead latency in the execution time estimation in its scheduling process.

4.6. Scheduling onto Spot Instances using bidding strategies

Another approach to utilize Spot Instances is proposed by Poola, Ramamohanarao and Buyya [66] to minimize cost within a deadline. Instead of analyzing the bid price history, this technique uses a bidding strategy to estimate a bid price based on three factors, namely, the price of the standard instance counterpart, the time margin allowable for using Spot Instances and the growth rate of the bid price toward the price of the standard instance. As long as the time margin between the deadline and the current time of the critical path is sufficiently large, the algorithm prefers the

cheapest Spot Instance to reduce cost. Otherwise, it selects the instance type that yields the lowest cost-performance ratio.

This technique also includes the virtual machine overhead latency in the execution time estimation and tries to avoid starting up a new virtual machine instance if a running one can accommodate the task being scheduled. Although the technique calculates the cost based on the EC2 hourly charge period, it does not use the charge period in its scheduling process.

4.7. BTS and PBTS algorithms

The "Balanced Time Scheduling" (BTS) algorithm is proposed by Byun et al. [45, 63] for scheduling workflow in hybrid cloud environment. This algorithm identifies the "schedulable duration" within which each task can execute as late as possible. The tasks with a smaller schedulable duration are given higher priority followed by the tasks that require more hosts to execute. Tasks are scheduled in order to reduce the number of resources used and to save the schedulable duration of other tasks to allow for subsequent flexibility. The schedule is further optimized in the "task redistribution" phase that reschedules tasks originally scheduled in the busiest time slot to an earlier and a later slots to reduce the number of resources required to accommodate the busiest time slot.

The BTS algorithm is later improved to the "Partitioned Balanced Time Scheduling" (PBTS) algorithm [19] to address the charge period of virtual machine and to elastically instantiate and terminate additional virtual machine instances during execution. The algorithm builds schedule iteratively based on charge period partitions (e.g. a partition of one hour according to Amazon EC2) so that the number of virtual machine instances in each partition can be adjusted at run-time. In the first phase, the number of machine instances is estimated based on workflow structure. The tasks that are able to finish within the next partition are then selected to be scheduled using the BTS algorithm. Also, to utilize a free remaining time slot of a partition that occurs during execution, PBTS assigns to the slot the task that can finish within the partition.

4.8. IC-PCP and IC-PCPD2 algorithms

The "IaaS Cloud Partial Critical Paths" (IC-PCP) and the "IaaS Cloud Partial Critical Paths with Deadline Distribution" (IC-PCPD2) algorithms are proposed by Abrishami, Naghibzadeh and Epema [23], extending their previous "PCP" algorithm in [30]. The IC-PCP algorithm determines a critical path and its deadline backward from the last task in the workflow. All the tasks in the critical path are assigned to the cheapest virtual machine instance that can complete them within the deadline so the data transfer can be avoided. The algorithm prefers choosing an instance with leftover time

of its charge period (as it incurs no additional cost) to launching a new instance. Similar process is repeatedly performed on the remaining paths in the workflow until all the tasks are scheduled. This algorithm has later been extended with privacy constraint into the MPHC algorithm [43].

The IC-PCPD2 algorithm, instead of determining critical paths, distributes the workflow deadline to all the tasks. Each task is then assigned to the cheapest virtual machine instance that can complete the task within its individual deadline and that has a leftover time from its previous charge period. Both algorithms assume that the execution of a workflow takes place in a single cloud availability zone (i.e. data center) thus the data transfer cost between virtual machines is ignored [23].

4.9. MER algorithm

The "Maximum Effective Reduction" (MER) algorithm is presented in [73, 77]. The algorithm takes a schedule generated by any other algorithm as its input to further optimizes the resource efficiency of the schedule. The technique uses task consolidation and resource consolidation to reduce the number of virtual machine instances with a small increase of makespan. Task consolidation starts from the resource that is assigned the last task in the original schedule. The tasks scheduled to this resource, where possible within a delay limit, are reassigned to other resources that cause the minimum increase of makespan. Afterward, the resources whose cores are underutilized are consolidated. Iteratively, the tasks assigned to the resources with the minimum cores used are reassigned to a resource with a higher utilization. Through these consolidations, this algorithm maximizes virtual machine utilization and achieves the best performance when a workflow has many short parallel tasks.

4.10. A multi-objective list scheduling approach

Fard et al. [25] proposes a list scheduling technique that can accommodate multiple scheduling criteria including makespan, cost, energy consumption and reliability. The possible value ranges of the criteria are presented to users so they can set the soft constraints along with the decision weight of each criterion. The algorithm builds a complete schedule by iteratively mapping the growing partial workflow graph globally using multi-objective optimization. The user's constraints are partitioned to each task, which are ranked according to the workflow precedent dependencies.

In each scheduling step, the immediate constraint values for the sub-graph are calculated and all the possible partial schedules are generated. The partial schedule that overwhelms the immediate constraint values to the greatest extent is selected for that step. If no such partial schedule is found, the algorithm finds the nearest partial schedule to the intermediate constraint values based on the

Euclidian distance, and selects the solution that overwhelms the nearest partial schedule to the greatest extent; if none exists, then the nearest partial schedule is selected.

4.11. SABA algorithm

The "security-aware and budget-aware workflow scheduling strategy" (SABA) is proposed by Zeng, Veeravalli, and Li [20] to schedule a workflow under a budget constraint while taking into account data security requirement. The technique defines the concept of "movable data" and "immovable data" to impose restrictions on data movement and duplication. For the mapping of tasks, the algorithm chooses a virtual machine instance that has the best performance-to-cost ratio when compared to the fastest instance type. The technique also introduces the consideration of security services such as encryption and decryption on the data, along with their time overheads as additions to execution time.

4.12. Scheduling with multi-level security requirement

To address a more complicated security requirement, Watson [42] proposes a technique that defines multi-level security requirement in workflow scheduling. Each task and data item is assigned with a rule that defines a security level requirement. The algorithm then finds all the possible schedules by assigning the tasks to the cloud providers that can satisfy the security rules of each task and its related data items. If many valid schedules are available, the schedule that incurs the least cost, calculated from compute cost, data transfer cost and storage cost, is selected as the final schedule.

4.13. Classification and comparison of scheduling techniques

The classification and comparison of the scheduling techniques based on our taxonomies are presented in Table 1, which also included the techniques that are not explained in this paper due to space limitation. For the scheduling criteria columns, the characters "h" and "s" respectively denote the hard and soft variations of each criterion. The symbol "+" in the execution model columns denotes the techniques that assume the workflow execution based on multiple cloud providers; only the techniques that clearly defines a model that reflects multiple cloud providers are assigned to this subgroup.

Table 1 Summary of cloud workflow scheduling techniques

Technique / Literature	Scheduling Criteria					Schedule generation			Task-resource mapping		
	Time	Cost	Reliability	Energy	Security	Constructive	Global	Hybrid	Single -Rigid	Multiple - Rigid	Multiple - Malleable
CTC-MC/MT [74]	h/s	s/h				✓			✓		
Pandey et al. [52]	s	s						✓	✓		
Barrett et al. [70]	h	h					✓		✓		
RC² [44]	h	s				✓			✓		
HCOC [18]	h	s				✓			✓		
DCP-C [65]	h	s				✓			✓		
Poola et al. [66]	h	s				✓			✓		
BTS [45]	h	s				✓					✓
PBTS [19]	h	s				✓					✓
IC-PCP/D2 [23]	h	s				✓			✓		
MER [73, 77]	h	s					✓		✓		
BMT and BMEMT [46]	h	s				✓			✓		
Kang et al. [72]	h	s					✓		✓		
Wang et al. [12]	s	s				✓			✓		
Bessai et al. [78]	s	s				✓			✓		
Wu et al. [79]	s	s					✓		✓		
Fard et al. [25]	s	s	s	s				✓	✓		
Chen and Zhang [80]	h	h	h				✓		✓		
SABA [20]	s	h			h (single)	✓			✓		
Watson [42]	s	s			h (multi)		✓		✓		
MPHC [43]	h	s			h (multi)	✓			✓		
Malawski et al. [11, 81]	h	h				✓			✓		

"h" and "s" denote hard and soft variations

Table 1 (Continued)

Technique / Literature	Execution model		Compute cost			Data transfer cost	Storage cost	Provisioning model			Task execution capacity		VM Startup latency
	Public cloud	Hybrid cloud	VM Count	VM Time Unit	Charge Period			On-demand	Unreliable Provisioning	Long-term Contracted	Single-task	Multi-task	
CTC [74]	✓		✓[1]			✓		✓			✓		
Pandey et al. [52]	✓+			✓		✓		✓			✓		
Barrett et al. [70]	✓			✓		✓		✓		✓	✓		
RC² [44]		✓	✓					✓			✓		

HCOC [18]		✓+		✓				✓				✓	
DCP-C [65]		✓		✓				✓	✓		✓		✓
Poola et al. [66]	✓			✓				✓	✓		✓		✓
BTS [45]		✓+		✓				✓			✓		
PBTS [19]		✓+			✓			✓			✓		
IC-PCP/D2 [23]	✓				✓			✓			✓		
MER [73]	✓		✓					✓				✓	
BMT and BMEMT [46]	✓+			✓				✓			✓		
Kang et al. [72]	✓				✓			✓				✓	✓
Wang et al. [12]	✓				✓			✓			✓		
Bessai et al. [78]	✓+			✓		✓		✓			✓		
Wu et al. [79]	✓+			✓		✓		✓[2]			✓[2]		
Fard et al. [25]	✓+			✓		✓	✓	✓			✓		
Chen and Zhang [80]	✓			✓				✓			✓		
SABA [20]	✓+			✓		✓		✓			✓		
Watson [42]		✓+		✓		✓	✓	✓[2]			✓[2]		
MPHC [43]		✓+			✓	✓	✓	✓			✓		
Malawski et al. [11, 81]	✓				✓			✓			✓		✓

"+" denotes multiple public cloud providers
[1]Cost is defined as price per task execution
[2]Resources are defined as services

5. Conclusion and Future Work

We present taxonomies of cloud workflow scheduling problems and techniques based on analysis of existing research literature. Our taxonomies aims to complement the taxonomies in [4], which classifies techniques in grid workflow scheduling, by adding new aspects unique to cloud computing and refining some existing ones. Several notable techniques and algorithms are reviewed and classified based on our taxonomies. We hope that the proposed taxonomies will provide a baseline for further development of cloud workflow scheduling techniques.

It is noticeable that almost every technique proposed so far has the assumption that resources are virtual machine instances (i.e. infrastructure-as-a-service). This is probably because scientific workflow applications are executed mainly on virtual machine instances preinstalled with necessary programs or software. However, cloud resources are not limited to only virtual machines.

As cloud computing evolves, more cloud services have been made available in platform-as-a-service and software-as-a-service levels in general (covering other "-as-a-Service" that have been invented in specific areas). Scientific workflows may soon, if not already, utilize other type of cloud resources to a greater extent [42]. Consequently, cloud workflow scheduling shall soon face with another challenge as the services in higher deployment levels may have different execution characteristics and cost models.

As for our future work, we are investigating a scheduling technique for scheduling parallel instances of a workflow in the cloud. Many of the existing grid and cloud workflow scheduling techniques usually need to know the structure of a workflow in order to estimate workflow

makespan and cost, and to find the critical path of the workflow [19, 23, 27]. Such techniques also require that a workflow is represented as directed acyclic graph (DAG) [1, 4, 14, 19, 82], leaving the loop structure to be addressed by a separate mechanism such as loop unrolling. This limitation is of concern when scheduling a workflow whose multiple instances (or multiple workflows) do not arrive at the same time [12, 15, 83, 84] making it hard to represent the whole structure. To avoid this limitation, the challenge when scheduling against a deadline or a budget lies in estimating the execution progress (e.g. the number of completed instances) without knowing the structure of the workflow.

6. Acknowledgements

This research is funded by A New Researcher Scholarship of CSTS, MOST by the Coordinating Center for Thai Government Science and Technology Scholarship Students (CSTS) of the National Science and Technology Development Agency under project ID SCH-NR2012-212.

7. References

[1] 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.

[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] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Generation Computer Systems,* vol. 25, pp. 599-616, 2009.

[4] M. Wieczorek, A. Hoheisel and R. Prodan, "Taxonomies of the Multi-Criteria Grid Workflow Scheduling Problem," in *Grid Middleware and Services*, ed, 2008, pp. 237-264.

[5] 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, pp. 237-256, 2009.

[6] 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.

[7] 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.

[8] L. Liu, M. Zhang, Y. Lin, and L. Qin, "A Survey on Workflow Management and Scheduling in Cloud Computing," in *Proceedings of the 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2014)*, Chicago, 2014, pp. 837-846.

[9] L. K. Arya and A. Verma, "Workflow scheduling algorithms in cloud environment - A survey," in *Proceedings of the 2014 Recent Advances in Engineering and Computational Sciences (RAECS)*, Chandigarh, India, 2014, pp. 1-4.

[10] Vijindra and S. Shenai, "Survey on Scheduling Issues in Cloud Computing," *Procedia Engineering,* vol. 38, pp. 2881-2888, 2012.

[11] M. Malawski, G. Juve, E. Deelman, and J. Nabrzyski, "Algorithms for cost- and deadline-constrained provisioning for scientific workflow ensembles in IaaS clouds," *Future Generation Computer Systems,* vol. 48, pp. 1-18, 2015.

[12] J. Wang, P. Korambath, I. Altintas, J. Davis, and D. Crawl, "Workflow as a Service in the Cloud: Architecture and Scheduling Algorithms," *Procedia Computer Science,* vol. 29, pp. 546-556, 2014.

[13] S. Smanchat and K. Viriyapant, "Identifying Information Requirement for Scheduling Kepler Workflow in the Cloud," *Procedia Computer Science,* vol. 29, pp. 1762-1769, 2014.

[14] H. Topcuoglu, S. Hariri and M. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE Transactions on Parallel and Distributed Systems,* vol. 13, pp. 260-274, 2002.

[15] 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.

[16] K. Liu, J. Chen, H. Jin, and Y. Yang, "A Min-Min Average Algorithm for Scheduling Transaction-Intensive Grid Workflows," in *Proceedings of the 7th Australasian Symposium on Grid Computing and e-Research (AusGrid 2009)*, Wellington, New Zealand, 2009, pp. 41-48.

[17] S. Smanchat and S. Sritawathon, "A scheduling algorithm for grid workflow using bottleneck detection and load balancing," *International Journal of Web Information Systems,* vol. 10, pp. 263-274, 2014.

[18] 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.

[19] 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.

[20] L. Zeng, B. Veeravalli and X. Li, "SABA: A security-aware and budget-aware workflow scheduling strategy in clouds," *Journal of Parallel and Distributed Computing,* vol. 75, pp. 141-151, 2015.

[21] W. Zheng and R. Sakellariou, "Budget-Deadline Constrained Workflow Planning for Admission Control," *Journal of Grid Computing,* vol. 11, pp. 633-651, 2013/12/01 2013.

[22] L. Zeng, B. Veeravalli and X. Li, "ScaleStar: Budget Conscious Scheduling Precedence-Constrained Many-task Workflow Applications in Cloud," in *Proceedings of the 2012 IEEE 26th International Conference on Advanced Information Networking and Applications (AINA)*, Fukuoka, 2012, pp. 534-541.

[23] 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.

[24] D. A. Menascé and E. Casalicchio, "A framework for resource allocation in grid computing," in *Proceedings of the 12th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS 2004)*, Volendam, The Netherlands, 2004, pp. 259-267.

[25] H. M. Fard, R. Prodan, J. J. D. Barrionuevo, and T. Fahringer, "A Multi-objective Approach for Workflow Scheduling in Heterogeneous Environments," in *Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2012)*, Ottawa, 2012, pp. 300-309.

[26] L. Grandinetti, O. Pisacane and M. Sheikhalishahi, "An approximate ?-constraint method for a multi-objective job scheduling in the cloud," *Future Generation Computer Systems,* vol. 29, pp. 1901-1908, 2013.

[27] 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*, Melbourne, 2005, pp. 140-147.

[28] J. Yu and R. Buyya, "Scheduling scientific workflow applications with deadline and budget constraints using genetic algorithms," *Sci. Program.,* vol. 14, pp. 217-230, 2006.

[29] N. Ranaldo and E. Zimeo, "Time and Cost-Driven Scheduling of Data Parallel Tasks in Grid Workflows," *IEEE Systems Journal,* vol. 3, pp. 104-120, 2009.

[30] S. Abrishami, M. Naghibzadeh and D. H. J. Epema, "Cost-Driven Scheduling of Grid Workflows Using Partial Critical Paths," *IEEE Transactions on Parallel and Distributed Systems,* vol. 23, pp. 1400-1414, 2012.

[31] J. Yu, R. Buyya and C. K. Tham, "QoS-based Scheduling of Workflow Applications on Service Grids," in *Proceedings of the 1st IEEE International Conference on e-Science and Grid Computing (e-Science'05)*, Melbourne, Australia, 2005.

[32] M. Wieczorek, S. Podlipnig, R. Prodan, and T. Fahringer, "Bi-criteria Scheduling of Scientific Workflows for the Grid," in *Proceedings of the 8th IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2008)*, Lyon, 2008, pp. 9-16.

[33] L. Zhao, Y. Ren and K. Sakurai, "Reliable workflow scheduling with less resource redundancy," *Parallel Computing,* vol. 39, pp. 567-585, 2013.

[34] 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.

[35] X. Wang, C. S. Yeo, R. Buyya, and J. Su, "Optimizing the makespan and reliability for workflow applications with reputation and a look-ahead genetic algorithm," *Future Generation Computer Systems,* vol. 27, pp. 1124-1134, 2011.

[36] Amazon Web Services. *Amazon EC2 Spot Instances*. Available: http://aws.amazon.com/ec2/spot-instances/ Accessed December 2014.

[37] A. Beloglazov, J. Abawajy and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing," *Future Generation Computer Systems,* vol. 28, pp. 755-768, 2012.

[38] J. J. Durillo, V. Nae and R. Prodan, "Multi-objective Workflow Scheduling: An Analysis of the Energy Efficiency and Makespan Tradeoff," in *Proceedings of the 13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2013)*, Delft, 2013, pp. 203-210.

[39] Y. Luo and S. Zhou, "Power Consumption Optimization Strategy of Cloud Workflow Scheduling Based on SLA," *WSEAS Transactions on Systems,* vol. 13, 2014.

[40] C. Zhang, E.-C. Chang and R. H. C. Yap, "Tagged-MapReduce: A General Framework for Secure Computing with Mixed-Sensitivity Data on Hybrid Clouds," in *Proceedings of the 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2014)*, Chicago, 2014, pp. 31-40.

[41] J. C. Mace, A. v. Moorsel and P. Watson, "The case for dynamic security solutions in public cloud workflow deployments," in *Proceedings of the 2011 IEEE/IFIP 41st International Conference on Dependable Systems and Networks Workshops (DSN-W)*, Hong Kong, 2011, pp. 111-116.

[42] P. Watson, "A multi-level security model for partitioning workflows over federated clouds," *Journal of Cloud Computing,* vol. 1, pp. 1-15, 2012.

[43] S. Sharif, J. Taheri, A. Y. Zomaya, and S. Nepal, "MPHC: Preserving Privacy for Workflow Execution in Hybrid Clouds," in *Proceedings of the 2013 International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT 2013)*, Taipei, 2013, pp. 272-280.

[44] 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.

[45] 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.

[46] T. A. L. Genez, L. F. Bittencourt and E. R. M. Madeira, "Workflow scheduling for SaaS / PaaS cloud providers considering two SLA levels," in *Proceedings of the 2012 IEEE Network Operations and Management Symposium (NOMS)*, Maui, USA, 2012, pp. 906-912.

[47] A. Abraham, H. Liu, C. Grosan, and F. Xhafa, "Nature Inspired Meta-heuristics for Grid Scheduling: Single and Multi-objective Optimization Approaches," in *Metaheuristics for Scheduling in Distributed Computing Environments*, ed, 2008, pp. 247-272.

[48] C. Szabo, Q. Z. Sheng, T. Kroeger, Y. Zhang, and J. Yu, "Science in the Cloud: Allocation and Execution of Data-Intensive Scientific Workflows," *Journal of Grid Computing,* vol. 12, pp. 245-264, 2014/06/01 2014.

[49] W.-N. Chen and J. Zhang, "An ant colony optimization approach to a grid workflow scheduling problem with various QoS requirements," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews,* vol. 39, pp. 29-43, 2009.

[50] Z. Wu, X. Liu, Z. Ni, D. Yuan, and Y. Yang, "A market-oriented hierarchical scheduling strategy in cloud workflow systems," *Journal of Supercomputing,* vol. 63, pp. 256-293, 2013/01/01 2013.

[51] S. Xue and W. Wu, "Scheduling Workflow in Cloud Computing Based on Hybrid Particle Swarm Algorithm," *TELKOMNIKA Indonesian Journal of Electrical Engineering,* vol. 10, pp. 1560-1566, 2012.

[52] 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)*, Perth, Australia, 2010, pp. 400-407.

[53] S. Smanchat, M. Indrawan, S. Ling, C. Enticott, and D. Abramson, "Scheduling Multiple Parameter Sweep Workflow Instances on the Grid," in *Proceedings of the 5th IEEE International Conference on e-Science (e-Science '09)*, Oxford, UK, 2009, pp. 300-306.

[54] M. Maheswaran, S. Ali, H. J. Siegel, D. Hensgen, and R. F. Freund, "Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems," in *Proceedings of the 8th Heterogeneous Computing Workshop (HCW '99)*, San Juan, 1999, pp. 30-44.

[55] H. Casanova, A. Legrand, D. Zagorodnov, and F. Berman, "Heuristics for scheduling parameter sweep applications in grid environments," in *Proceedings of the 9th Heterogeneous Computing Workshop (HCW 2000)*, Cancun, 2000, pp. 349-363.

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

[57] 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.

[58] S. Ostermann, R. Prodan and T. Fahringer, "Dynamic Cloud provisioning for scientific Grid workflows," in *Proceedings of the 11th IEEE/ACM International Conference on Grid Computing (GRID 2010)*, Brussels, 2010, pp. 97-104.

[59] Amazon Web Services. *Amazon EC2*. Available: http://aws.amazon.com/ec2/ Accessed November, 2014.

[60] 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.

[61] D. Ardagna, E. D. Nitto, P. Mohagheghi, S. Mosser, C. Ballagny, F. D'Andria, G. Casale, P. Matthews, C.-S. Nechifor, D. Petcu, A. Gericke, and C. Sheridan, "MODAClouds: A model-driven approach for the design and execution of applications on multiple Clouds," in *Proceedings of the 2012 ICSE Workshop on Modeling in Software Engineering (MiSE)*, Zurich, 2012, pp. 50-56.

[62] R. d. C. Coutinho, L. M. A. Drummond, Y. Frota, and D. de Oliveira, "Optimizing virtual machine allocation for parallel scientific workflows in federated clouds," *Future Generation Computer Systems,* vol. 46, pp. 51-68, 2015.

[63] E. Byun, Y. Kee, E. Deelman, K. Vahi, G. Mehta, and J. Kim, "Estimating Resource Needs for Time-Constrained Workflows," in *Proceedings of the 4th IEEE International Conference on eScience (e-Science '08)*, Indianapolis, 2008, pp. 31-38.

[64] M. Mao and M. Humphrey, "Auto-scaling to minimize cost and meet application deadlines in cloud workflows," in *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, Seattle, Washington, 2011, pp. 1-12.

[65] S. Ostermann and R. Prodan, "Impact of Variable Priced Cloud Resources on Scientific Workflow Scheduling," in *Euro-Par 2012 Parallel Processing*. vol. 7484, C. Kaklamanis*, et al.*, Eds., ed: Springer Berlin Heidelberg, 2012, pp. 350-362.

[66] D. Poola, K. Ramamohanarao and R. Buyya, "Fault-tolerant Workflow Scheduling using Spot Instances on Clouds," *Procedia Computer Science,* vol. 29, pp. 523-533, 2014.

[67] Google. *Google Compute Engine*. Available: https://cloud.google.com/compute/ Accessed January, 2015.

[68] Microsoft. *Microsoft Azure: Virtual Machines*. Available: http://azure.microsoft.com/en-us/services/virtual-machines/ Accessed January, 2015.

[69] Amazon Web Services. *AWS Global Infrastructure*. Available: http://aws.amazon.com/about-aws/global-infrastructure/ Accessed December, 2014.

[70] 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)*, Lugano, 2011, pp. 83-90.

[71] Amazon Web Services. *Amazon S3*. Available: http://aws.amazon.com/s3/ Accessed January, 2015.

[72] D.-K. Kang, S.-H. Kim, C.-H. Youn, and M. Chen, "Cost adaptive workflow scheduling in cloud computing," in *Proceedings of the 8th International Conference on Ubiquitous Information Management and Communication*, Siem Reap, Cambodia, 2014, pp. 1-8.

[73] Y. C. Lee, H. Han and A. Y. Zomaya, "On Resource Efficiency of Workflow Schedules," *Procedia Computer Science,* vol. 29, pp. 534-545, 2014.

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

[75] 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.

[76] M. Rahman, S. Venugopal and R. Buyya, "A Dynamic Critical Path Algorithm for Scheduling Scientific Workflow Applications on Global Grids," in *Proceedings of the IEEE International Conference on e-Science and Grid Computing*, Bangalore, 2007, pp. 35-42.

[77] Y. C. Lee, H. Han, A. Y. Zomaya, and M. Yousif, "Resource-efficient workflow scheduling in clouds," *Knowledge-Based Systems,* 2015.

[78] K. Bessai, S. Youcef, A. Oulamara, C. Godart, and S. Nurcan, "Bi-criteria Workflow Tasks Allocation and Scheduling in Cloud Computing Environments," in *Proceedings of the 2012 IEEE 5th International Conference on Cloud Computing*, Honolulu, 2012, pp. 638-645.

[79] Z. Wu, Z. Ni, L. Gu, and X. Liu, "A Revised Discrete Particle Swarm Optimization for Cloud Workflow Scheduling," in *Proceedings of the 2010 International Conference on Computational Intelligence and Security (CIS)*, Nanning, 2010, pp. 184-188.

[80] W.-N. Chen and J. Zhang, "A set-based discrete PSO for cloud workflow scheduling with user-defined QoS constraints," in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC 2012)*, Seoul, 2012, pp. 773-778.

[81] M. Malawski, G. Juve, E. Deelman, and J. Nabrzyski, "Cost- and deadline-constrained provisioning for scientific workflow ensembles in IaaS clouds," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, Salt Lake City, Utah, 2012, pp. 1-11.

[82] Z. Shi and J. J. Dongarra, "Scheduling workflow applications on processors with different capabilities," *Future Generation Computer Systems,* vol. 22, pp. 665-675, 2006.

[83] 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.

[84] C. Enticott, T. Peachey, D. Abramson, D. Gavaghan, A. Bond, D. Elton, E. Mashkina, C.-Y. Lee, and G. Kennedy, "Electrochemical Parameter Optimization using Scientific Workflows," in *Proceedings of the 6th IEEE International Conference on e-Science*, Brisbane, 2010, pp. 324 - 330.

Figure Captions

Table Captions

Figure 1

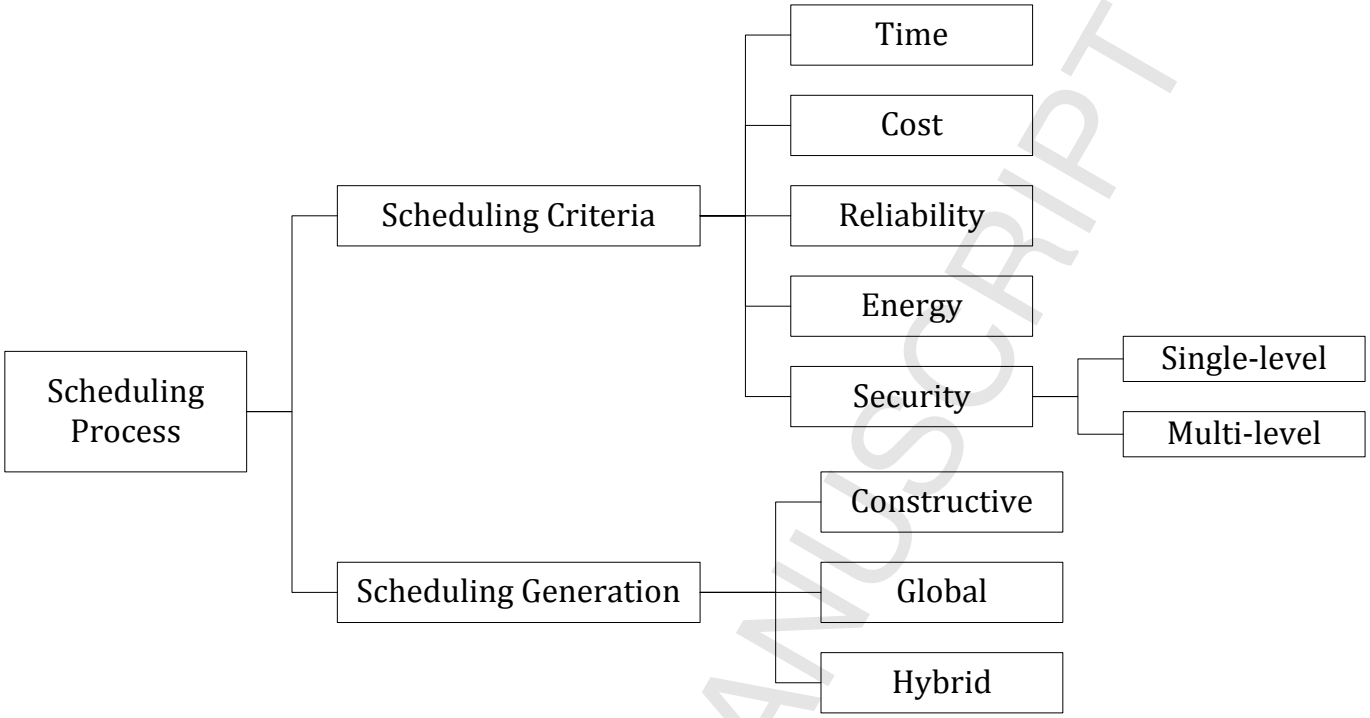


Figure 2

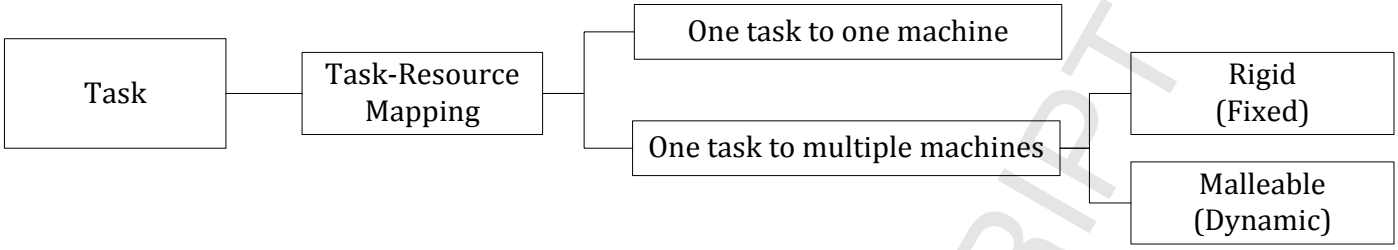


Figure 3

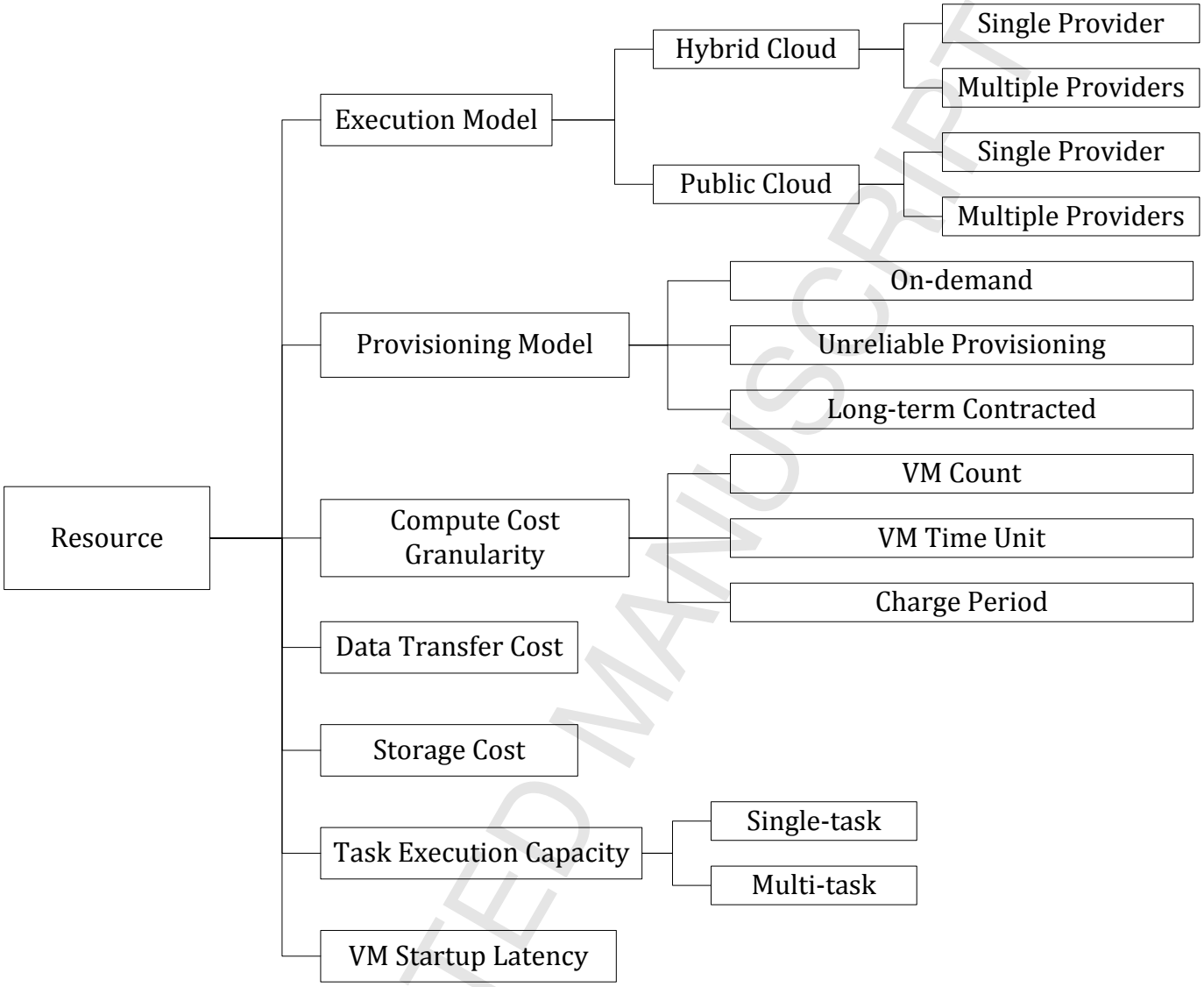


Table 1

Technique / Literature	Scheduling Criteria					Schedule generation			Task-resource mapping		
	Time	Cost	Reliability	Energy	Security	Constructive	Global	Hybrid	Single -Rigid	Multiple - Rigid	Multiple - Malleable
CTC-MC/MT [74]	h/s	s/h				✓			✓		
Pandey et al. [52]	s	s						✓	✓		
Barrett et al. [70]	h	h					✓		✓		
RC² [44]	h	s				✓			✓		
HCOC [18]	h	s				✓			✓		
DCP-C [65]	h	s				✓			✓		
Poola et al. [66]	h	s				✓			✓		
BTS [45]	h	s				✓					✓
PBTS [19]	h	s				✓					✓
IC-PCP/D2 [23]	h	s				✓			✓		
MER [73, 77]	h	s					✓		✓		
BMT and BMEMT [46]	h	s				✓			✓		
Kang et al. [72]	h	s					✓		✓		
Wang et al. [12]	s	s				✓			✓		
Bessai et al. [78]	s	s				✓			✓		
Wu et al. [79]	s	s					✓		✓		
Fard et al. [25]	s	s	s	s				✓	✓		
Chen and Zhang [80]	h	h	h				✓		✓		
SABA [20]	s	h			h (single)	✓			✓		
Watson [42]	s	s			h (multi)		✓		✓		
MPHC [43]	h	s			h (multi)	✓			✓		
Malawski et al. [11, 81]	h	h				✓			✓		

"h" and "s" denote hard and soft variations

Technique / Literature	Execution model		Compute cost			Data transfer cost	Storage cost	Provisioning model			Task execution capacity		VM Startup latency
	Public cloud	Hybrid cloud	VM Count	VM Time Unit	Charge Period			On-demand	Unreliable Provisioning	Long-term Contracted	Single-task	Multi-task	
CTC [74]	✓		✓[1]			✓		✓			✓		
Pandey et al. [52]	✓+			✓		✓		✓			✓		
Barrett et al. [70]	✓			✓		✓		✓		✓	✓		
RC[2] [44]		✓	✓					✓			✓		
HCOC [18]		✓+		✓				✓				✓	
DCP-C [65]		✓		✓				✓	✓		✓		✓
Poola et al. [66]	✓			✓				✓	✓		✓		✓
BTS [45]		✓+		✓				✓			✓		
PBTS [19]		✓+			✓			✓			✓		
IC-PCP/D2 [23]	✓				✓			✓			✓		
MER [73]	✓		✓					✓				✓	
BMT and BMEMT [46]	✓+			✓				✓			✓		
Kang et al. [72]	✓				✓			✓				✓	✓
Wang et al. [12]	✓				✓			✓			✓		
Bessai et al. [78]	✓+			✓		✓		✓			✓		
Wu et al. [79]	✓+			✓		✓		✓[2]			✓[2]		
Fard et al. [25]	✓+			✓		✓	✓	✓			✓		
Chen and Zhang [80]	✓			✓				✓			✓		
SABA [20]	✓+			✓		✓		✓			✓		
Watson [42]		✓+		✓		✓	✓	✓[2]			✓[2]		
MPHC [43]		✓+			✓	✓	✓	✓			✓		
Malawski et al. [11, 81]	✓				✓			✓			✓		✓

"+" denotes multiple public cloud providers
[1]Cost is defined as price per task execution
[2]Resources are defined as services

Biographies of all authors ordered according to the author list in the manuscript.

Sucha Smanchat

Sucha Smanchat is currently a lecturer at the Faculty of Information Technology, King Mongkut's University of Technology North Bangkok, Thailand. He obtained his PhD degree at Monash University in Melbourne, Australia. During his study, he is involved with the development of a prototype scheduler for Nimrod/K system. His current research interests are in cloud workflow scheduling techniques, MapReduce scheduling and data mining as cloud services.

Kanchana Viriyapant

Kanchana Viriyapant is currently a lecturer as well as a student at the Faculty of Information Technology, King Mongkut's University of Technology North Bangkok (KMUTNB), Thailand. She obtained a master's degree from KMUTNB and is now pursuing her PhD in cloud workflow scheduling area. Apart from her PhD topic, her expertise also covers data mining techniques.

Sucha's Photo

Click here to download high resolution image

Kanchana's Photo
Click here to download high resolution image