Journal of Ambient Intelligence and Humanized Computing (2021) 12:7621–7636
https://doi.org/10.1007/s12652-020-02483-0

ORIGINAL RESEARCH

Scheduling multiple scientific workflows using containers on IaaS cloud

P. Rajasekar[1] · Yogesh Palanichamy[1]

Received: 4 May 2020 / Accepted: 14 August 2020 / Published online: 26 August 2020

Abstract
With the adaptable paradigm of cloud computing and obtainable of data accumulated from largely high-powered scientific devices, workflows have turn into an occurring aim to execute considerable scientific advances at an enhanced speed. Occurring Workflow as a Service (WaaS) frameworks provide scientists an effortless, simply accessible and cost-efficient manner of using their applications from anywhere and at anytime in the cloud. They are multitenant platforms and are developed to handle the execution of heterogeneous workflows continuous workload. To fulfill this, they utilize the compute, network and storage services provided by Infrastructure as a Service (IaaS) vendors. Therefore, at any considerable particular moment, a WaaS framework should be proficient of effectively schedule these continuous workload of workflows with various features and quality of service (QoS). Therefore, we propose a strategy of scheduling and resource provisioning planned particularly for WaaS platforms. The algorithm is dynamic and scalable to adjust to improve in the workload and platform. It supports containers to deal the inefficiency of resource utilization and targets to reduce the overall execution cost of infrastructure resources as fulfilling each single workflow deadline constraint. To our information, this approach that explicitly deals VM sharing in the subject of WaaS by devising the utilization of containers in the heuristics of scheduling and resource provisioning. Our experimental results shows its responsiveness to the uncertainties of the environment, its potential to achieve deadlines, and its cost-effectiveness when compared to other recent algorithms.

Keywords Cloud computing · Workflow as a service · Scheduling · Resource provisioning · Deadline and cost minimization

1 Introduction

A group of computational tasks with dependencies are described as workflows. Workflows are a familiar application model is used in the field of computational science. They allow the examination of data in an organized and distributed way and have been used successfully to produce extraordinary scientific improvement in different specialization such as physics, biology, astronomy and medicine (Gil et al. 2007). Their significance is attracted in current big data span as they provide a productive way of execution and knowledge extraction from the data given by more powerful equipment's such as gravitational wave detectors, telescopes and particle accelerators. Therefore, it is usual for workflows to be a wide-range data and resource intensive model that are used on distributed platforms with the aim of generate solutions within a valid amount of time.

The occurrence of cloud computing has carried with it various benefits for the usage of scientific workflows. Especially, cloud infrastructure as a service (IaaS) permit workflow handling approach (WHA) to approach an infinite pool of resources that can be obtained, modified, and used as demanded virtually and are billed on pay-as-you-go basis. IaaS vendors provide compute resources virtually called virtual machines (VMs) for rent. They possess a pre-described CPU, storage, memory, and bandwidth capacity and the bundles of heterogeneous resource (i.e., VM types) are accessible at different cost. They can be able to elastically obtain and make free and are commonly charged per time unit. As VMs achieve the compute ability, IaaS also provide network

✉ P. Rajasekar
rajasekar@auist.net

Yogesh Palanichamy
yogesh@annauniv.edu

[1] Department of Information Science and Technology, College of Engineering, Anna University, Guindy, Chennai, Tamilnadu, India

and storage services, offering the demanded infrastructure for the workflow applications execution.

Scheduling approaches customized for workflow applications are significant in considering use of the advantages provided by clouds and they have been extensively analyzed in late years. To succeed this, they involve not only to prioritize on the mapping of task and resource but also on planning the type and number of resources to utilize throughout the workflow execution (i.e. provisioning of resource). Most existing strategies prioritize on making scheduling and resource provisioning decisions for a single workflow application. They believe resource models and application in which only one user presents a single instance of workflow for performance to a WHA. The WHA is then accountable for provisioning the demanded resources and scheduling the tasks to them hence that the execution of single workflow is done within the constraints of quality of service (QoS). As this is a reasonable design, as the support of cloud computing turns to be popular in the middle of scientific group, new applications types are developing.

Especially, workflow as a service (WaaS) is an arising notion in which workflows execution is provided as a service to scientific community. WaaS would divided as a presenting either at the software as a service or platform as a service model as vendors use compute, network, and storage resources provided by IaaS providers to satisfy the requirements addressed to a multitenant WHA. Workflows presented to that WHA associated with various users and are not certainly linked to each user; they would differ in size, structure, application, input data and the requirement of QoS in the middle of other characteristics. Therefore, schedulers need to be capable of execute a bundle of workflows with various structure that are arriving continuously for execution (in the absence of believing the type and number of workflows are studied in before). A sketch of a WaaS environment is shown in Fig. 1. Platforms knowing this service type are starting to come out in the publication. For instance, Filgueira et al. (2016) suggest a data intensive workflow application as a resource type that permits the simple composition and usage of stream based applications of workflow on cloud environment utilizing containers. Likewise, Skyport (Gerlach et al. 2014) is a processing environment having the ability of handling the multiple workflows execution in cloud by using containers to deal the deployment of software issues and inefficiency of resource utilization. Additional examples contain the middleware defined by Esteves and Veiga (2016) and the structure shown by Wang et al. (2014).

Workflows are usually formed of various types of tasks. In experimental expression, same type of tasks execute the same software procedure; that is, they do the same procedure of computation possibly on various sets of data. This represents that various types of tasks require various software procedures for their performance. Virtualization permits for the processing nature of these tasks to be simply tailored. For example, hardware level virtualization would utilized in the manner that the operating systems, directory structures and software packages, in the middle of others, able to be tailored for a particular task and saved as an image of VM. This image able to be simply utilized to use VMs having the ability of processing the tasks. This is the version examined by the most occurring workflow scheduling approaches for IaaS cloud. They prioritize on efficient way for renting and freeing VMs with the aim of satisfy a group of QoS requirements and in common reason that all VMs would utilized utilizing a single image of VM that includes all of the required software to run any task from any workflow. This expectation is realistic and valid when examining single workflow scheduling but not when multiple workflow (i.e., belong to different users) scheduling.

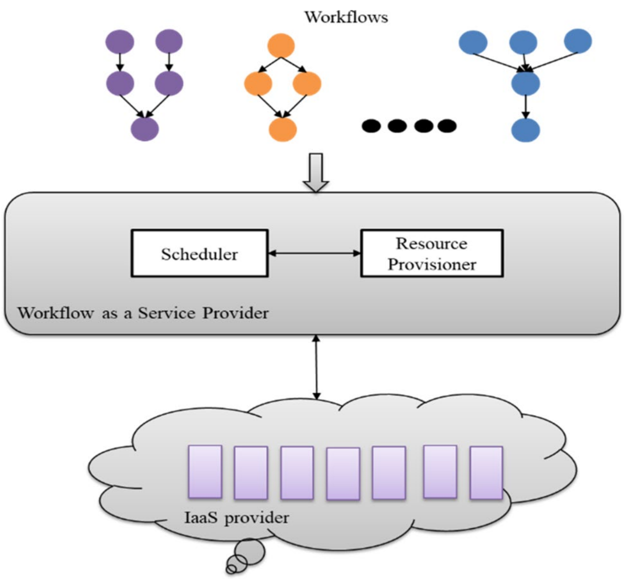


Fig. 1 The system architecture of our proposed approach

The important explanation for this is the incapable of customizing a single image of VM to encourage the execution of various tasks from various workflows (e.g., examine the image capacity and the inability among software programs demanded by various tasks from various workflows). WaaS frameworks able to be adopt various approaches to overcome this problem. An option is to provide dedicated VMs for each workflow (multiple workflows) for achieving each workflow QoS requirements such as deadline, but an opportunity is to run each independent workflow on its specific group of dedicated VMs or a group of associated workflows on their specific dedicated VMs would end in an inefficient utilization of resource and huge costs. Another opportunity that deal this problem is to integrate the utilization of containers and VMs, which are made of operating system virtualization. Container permit applications to be deployed and formed by offering a virtual nature that contain its particular network space, block I/O, memory and CPU. By permitting each workflow or task to have a relative container image, a VM would re-utilized to execute tasks corresponding to different workflows by deploying the relative container when a task is ready to execute on that VM. In this manner, maximize the resource utilization by minimize the idle time slots wastage on rented VMs.

Also using a well-planned VM sharing type, algorithms customized for WaaS frameworks would be adaptable because they have no information on the approaching workflows. They would also able to be changed in size and having the ability of generating decisions efficiently as the total tasks that demand to be executed at any particular moment. Other significant characteristic that would be assumed into examination is the effective auto-scaling and VMs management with the aim of improving their usage as a cost-effective technique as still can fulfill the QoS conditions of each individual workflow. This will possibly end in lesser cost for users and more profit for vendors. Lastly, algorithms would also deal usual challenges acquired from the resource type provided by cloud such as resource heterogeneity, resource abundance, unpredictability acquired from performance degradation, delays of VM provisioning and billing models.

In reply to these demands, we propose SRPSM, A Scalable Resource Provisioning and Scheduling algorithm for Multiple workflows developed for WaaS frameworks. It examines containers to deal inefficiency of resource utilization and targets to reduce the complete cost of renting resources as satisfying the deadline constraint of each individual workflow approaching continuously for execution. Even though there are several existing approaches developed for multiple workflow scheduling, they either implicitly or explicitly believe that each individual workflow in any situations (i.e., same workflow but differ in number of tasks) has its particular allocated resources. To our information, this approach that clearly deals VM sharing in the subject of WaaS by devising the utilization of containers in the heuristics scheduling and resource provisioning. Moreover, the algorithm is scalable and adjustable and our experimental results show its reactions to environmental unpredictability's, its potential to achieve the deadlines and its cost-effectiveness when compared to other algorithms.

2 Related work

Most of the algorithms in the publications prioritize on optimizing the single workflow execution based on its specific QoS needs. Therefore, the resources are utilized specifically for the single workflow related to single user for execution. The majority of algorithms contains as targets reducing the overall execution cost as satisfying a deadline limitation. Examples contain (Abrishami et al. 2013; Calheiros and Buyya 2013; Deldari et al. 2017; Liu et al. 2016) and approach proposed by Dziok et al. (2016). To succeed this, they possess strategies on the spot to elastically obtain and free resources and prioritize mainly on reutilizing the inactive time periods of provisioned VMs when achievable with the aim of maximize the utilization of resources and protect some cost. Anyway, the deadline limitation strategies and dependencies among tasks represent that unutilized time periods cannot be completely removed.

Several workflow applications are comprised of interconnected workflow groups called as ensembles (Deelman et al. 2008; Maechling et al. 2007; Vöckler et al. 2011). These workflows are linked together as their integrated performance generates desired output (Malawski et al. 2015). There are some scheduling approaches developing for this model of applications in the publication (Bryk et al. 2016; Chai 2020; Chhabra et al. 2020; Gupta et al. 2020; Jiang et al. 2015; Pietri et al. 2013; Rajan 2020; Thennarasu et al. 2020; Yin et al. 2020). They vary from the output given in this material in three aspects. Initially, the requirements of QoS are not described for each individual workflow, but instead for the whole ensemble. Therefore, the algorithms are usually examined and incline to have this in the scheduling targets. Secondly, the instance of number of workflows is usually studied in before and the scheduling heuristics would utilize this when organizing the performance of workflow tasks. Lastly, the same workflows in an ensemble, representing they possess a same structure but vary in input data and size.

Yu and Shi (2008) examined an application type same to the one deals in this material and offered an algorithm for scheduling the applications of multiple workflows presented from various user at different periods. Anyway, their output is customized for cluster infrastructure and it believes a static number of resources that are quickly accessible. In addition to, the algorithm's goal is to reduce the makespan of every single workflow rather than satisfying the constraint of deadline. Although these variations, we examine their project related because the authors not only find the requirements for such type, even in cluster infrastructures, but also find some significant issues and features that need to be examined despite distributed environment such as the requirement for the algorithm to be adaptable and the significance of examining the entire resource utilization from the perspective of resource management.

Some works examine multiple workflows scheduling in cloud. For example, the work done by Jiang et al. (2011), anyway, the application type they examine is varied to the one presented in this material because they believe the type and number of workflows are studied before and that whole workflows are presented for performance at the same period. In addition to, their approach does not examine deadlines nor cost because its main target is to increase the resource utilization. Another approach is done by Stavrinides and Karatza (2015), it is formed on a list scheduling strategy and their application type is same to the one addressed in this material but varies in the information that they examine the standard of the data generated by every individual workflow as portion of the QoS needs. Likewise, Xu et al. (2009) present a heuristic with various objectives as they examine a budget limitation and as portion of the QoS requirements. Lastly, Chen et al. (2015) propose an algorithm that contains the similar application type and scheduling targets as our presented output. Anyway, their proposed output as well as earlier mentioned outputs, vary from our strategy in a key characteristic. They believe a limited group of heterogeneous VMs that is accessible all over the whole lifespan of the procedure and therefore do not examine the problem of resource provisioning within abundant and elastic resources.

On the opposite, Dyna (Zhou et al. 2015), is a scheduling approach developed for IaaS with auto-scaling characteristics to dynamically assigned and unassigned VMs formed on the tasks' current status. It assignments by choosing VM categories for every workflow task formed on a search of a star so that the overall cost is reduced. Anyway, it varies from our strategy because it provides guarantees of probabilistic deadline and examines VMs cost within two different types: static (e.g., on-demand instances of amazon) and dynamic VMs (e.g., spot instances of amazon). In addition to, even though the policy of VM sharing is not clearly defined in the material, it would be concluded that a type in which same types of workflows only shared the VMs. This formed on the system type examined by the authors and the consideration which is carried out with a workload made up of the same kind of workflow but with various total number of tasks.

SCS (Mao and Humphrey 2011) and WPPDS (Shi et al. 2014) are also having the ability of scheduling multiple workflows in cloud with an auto-scaling feature. SCS first generates a resource provisioning scheme formed on a heuristic of global optimization and then improves it at processing time to react to unexpected delays that were disregarded for. Anyway, the improvement of the resource provisioning scheme is achieved by processing the global optimization procedure for the leftover tasks every time schedule a task. This makes a high overhead and limits its scalability in terms of the processing number of tasks. WPPDS on the next side examines a budget cost for each workflow deadlines and whole workload and its aim is to complete as multiple workflows as achievable with the considered budget.

Wang et al. (2014) presented a structure for a WaaS platform beside with four scheduling algorithms based on heuristic: static, scalable, greedy and adaptive. They vary from our approach because they only permit VMs to be shared among same workflow type of tasks but not among tasks belonging to various workflows. In addition to, the targets of their presented approaches are to reduce the cost and makespan and they do not examine data transfer times and VM provisioning delays. It is useful observing that out of the studied approaches, this is the only material that clearly describes the sharing policy of a VM. Remaining approaches either naively believe that any workflow task can be used on any of the accessible VMs or not succeed to adequately describe their type of application.

Asterism DIaas (Filgueira et al. 2016) and Skyport (Gerlach et al. 2014) are additional examples of WaaS platforms. They are related to this assignment because they find the advantages and requirement of utilizing containers but prioritize on how they would be utilized to bundle workflow tasks and the benefits of using them on VMs which are already provisioned so that these are having the ability of processing any tasks belong to any workflow. They do not prioritize on the problems of scheduling and resource provisioning deals in the material. Esteves and Veiga (2016) also describe a framework of prototypical middleware that represents the idea of a WaaS platform and deal problems such as description of workflow, WHA incorporation, resource allocation, and cost type. Their assignment prioritizes on workflows for incremental and continuous processing of data. Even though the authors highlight on the plan of a WaaS framework, their presented approach prioritize on a single workflow.

Finally, Rodriguez and Buyya (2017a, b) defined the billing price of cloud resource model. There are three types, first one is hourly billing (on demand static instance), second one is hourly billing (spot instance or dynamic instance) and final one is minute billing (on demand static instance). All of the surveyed approaches are used mostly hourly billing scheme for single and multiple workflow scheduling, except Zhou et al. (2015) used both static and dynamic instances. Any utilization of both instances used in partial is charged as whole period. For example, for rent 60 min, if a VM is used 62 min, the user have to pay double period of 60 min, that is, 120 min even if a VM is used only 25 min, the user will pay for 60 min. On the other hand, on demand instance of spot instance which is offered inconsistent cost during the billing period. For example, if rent a spot instance for 60 min, during its lease time either which cost is lower or very higher compare to on demand static instance and eventually charges more cost than on demand static instance. In addition to, it can be terminated by the provider at any time.

There are two disadvantages in partial utilization, first one, a user paid full hour price if used an instance partially or fully. Next one, instance utilization rate is calculated based on hour on hourly billing, therefore that makes low instance utilization rate. So these both instances are not completely adaptable for cost reduction and instance utilization to users. In WaaS platform, workflows are continuously arriving for execution and our aim is tend to less cost for users with high utilization rate and more profit for providers.

So, user utilize the suitable billing model to save the cost and increase the utilization rate. As a result, provider gain more profit. Therefore we define that with additional problem in the next section as well as based on the reason for the selection of suitable billing model for WaaS platform.

3 Motivation

3.1 Partial issue problem

Jin et al. (2014) findings, instances in pay per use pricing for the case of on demand VMs in Ec2, are suggested mostly for the applications with short workload, also that cannot be delayed (short jobs). These VMs are every time considered hourly, yet users with short jobs have to pay complete hour price even their jobs consumed small portion of resources. This is known as partial wastage problem. Hence, they evaluated the resource utilization instance time on one month Google instance traces of users that is depicted in Fig. 2. It demonstrates that the large number of Google users (42.40% users) utilized the instances only ($<20\%$), which made the crucial problem for the user. The reason of this problem has explained in previous section by the example of hourly VM charging cost. Also, this partial usage noticed in many research articles (Maechling et al. 2007; Malawski et al. 2015; Mao and Humphrey 2011) in cloud and it is a non-negligible one for cloud users. This partial issue in cloud resource is motivated to do a research carry out in this paper, and our aim is to overcome the issue of partial wastage to save the cost and increase the utilization for users.

Therefore, we consider the minute billing model for WaaS platform to reduce some amount of overall cost of infrastructure and increase the utilization than on demand static and dynamic instances. For example, the advantage of minute billing period is, if a VM is leased 60 min but used 85 min, user charged only for 85 min or if a VM is leased for 60 min but used 25 min, user charged only for 25 min. Hence, the minute billing scheme is more adaptable for jobs and reduce some amount of cost plus increased the utilization rate than hourly billing scheme of static and dynamic pricing. Moreover, it gives more profit for providers, because pay per minute VM offers low cost to users to submit more workflows therefore it offers more profit for IaaS providers.

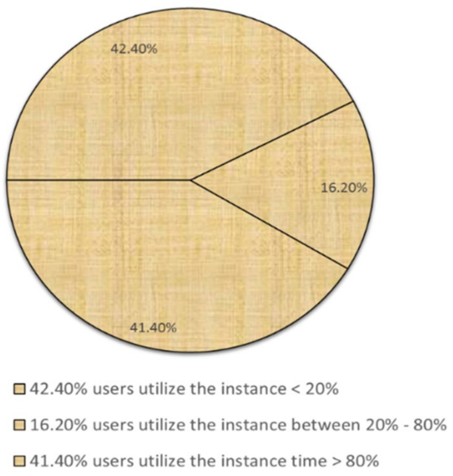


Fig. 2 Partial usage issue

4 Application and models of resource

This assignment is developed to schedule an uninterrupted bundle of multiple workflows presented by scientists to a WaaS vendor. The workflows probably contain various features such as application model, total number of tasks, I/O data and deadline limit. The WaaS vendor rents resources from an IaaS provider to satisfy the users' requirements and its aim is to reduce the overall cost of leasing resources as satisfying the deadline limitation of each individual workflow application submitted.

Scientific workflows are represented as directed acyclic graph (DAGs); that is, graphs in relation to guided edges and in relation to no cycle's dependencies, for example a sample scientific workflow is depicted in Fig. 3. At any considerable particular moment, there is a group $W=\{W_1, W_2,\ldots,W_n\}$ of scientific workflows that demand to be scheduled. In terms of structure, a workflow w is comprised of a group of tasks $T=\{t_1, t_2,\ldots,t_n\}$ and edges E. An edge $e_{ij}=(t_i, t_j)$ presents if at is a dependency in the middle of tasks t_i and t_j, mean task t_i is a parent of t_j and task t_j is a child of t_i. Found on this, child tasks cannot start its process until its all parent tasks have completed their execution. Lastly, each individual workflow is connected with a deadline dl_w, described as a time constraint for the workflow execution and a container cn_w that includes all the software and libraries demanded to run any workflow tasks.

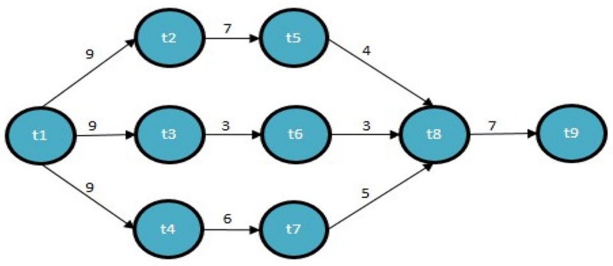


Fig. 3 An example of single workflow and its dependency with the value of data transfer time in-between parent and child tasks and archs

We examine a type in which workflow tasks run in containers which one after the other are used on VMs. A container would be used on VM at whatever time with a time of provisioning delay $prov_{cn}$. This delay links to the span it needs to process of container image downloading from the general storage approach such as Amazon S3 and put it on the VM. We examine a type where just one container would be used on the VM at a considerable particular moment and therefore, we believe containers access the same bandwidth capacity and CPU belong to the VM. We believe the execution of tasks would be initiated by global schedulers by communicating standard signals towards containers by make use of the command Docker exec. Therefore, various tasks would be processed one after the other on one container in the absence of redeployment.

We accept a pay per minute model where VMs are charged per billing minute τ. We examine a single data center and a single cloud provider. In this manner, reduce the network delays and eliminate the cost of intermediate data transfer. Lastly, we force no constraint on the total number of VMs that would be rented from the vendor.

The IaaS vendor presents a group of VM categories $VMT=\{vmt_1, vmt_2,\ldots,vmt_n\}$ with various configurations and prices. VMs categories are described in terms of their bandwidth capacity b_{vmt}, CPU processing capacity Pr_{vmt}, and cost per unit time c_{vmt}. A measure of average of their provisioning delay $prov_{vm}$ is also contained as portion of their description. Execution time, ET_t^{VMT}, of each task on each VM category is present to the scheduler. Various performance estimation strategies would be applied to acquire this value, in our strategy we compute it by estimate of the task size and CPU processing capacity of the VM category with included the percentage of performance variation is put in Eq. (1). Moreover, this obtained value of task execution time by this estimation method is not 100% correct to succeed its targets.

$$ET_t^{VMT} = TS_t/Pr_{vmt} \tag{1}$$

Found on the description of the workflow tasks executed by Juve et al. (2013) and the VM categories presented by Amazon ec2 cloud, we believe that all VM categories contain adequate memory to run any workflow tasks. Anyway, as a future assignment we will examine expanding the algorithm to contain strategies that guarantee tasks are allocated to VMs with adequate memory to run them. Moreover, we believe VMs contain single core for scheduling objectives and therefore are just efficient of executing just one task at a time.

We describe the data sharing in the middle of tasks to happen via general storage approach such as Amazon s3. In this manner, the outputs of tasks' store in the general storage and as similar to retrieve their inputs. We believe the general storage (GS) with adequate capacity and writing and reading speed of GS_w and GS_r separately. The time it needs to move and write the output data d from the vmt type of VM into the general storage is put in Eq. (2).

$$N_{d,vmt}^{output} = (d/b_{vmt}) + (d/GS_w) \tag{2}$$

Likewise, the time it needs to move and retrieve the input data d from the general storage into the VM type of vmt is put in Eq. (3).

$$N_{d,vmt}^{input} = (d/b_{vmt}) + (d/GS_r) \tag{3}$$

We admit that features such as multitenancy, heterogeneity, non-virtualized hardware, and virtualization in IaaS cloud makes performance variability in resources (Juve et al. 2010, 2013; Kouki and Ledoux 2013; Schad et al. 2010; Wang et al. 2013). Especially, (Schad et al. 2010) findings, performance degradation in cloud network resources when their maximal attainable performance being found on the bandwidth capacity described by the provider. Eventually, this end in a degradation in data transfer and guides to the execution time delay. Moreover, we do not believe there is a further performance degradation due to container deployment in this experiment (Tommaso et al. 2015; Felter et al. 2015).

As put in Eq. (4), calculate the total processing time of task t TPT_t^{VMT} on a VM of category vmt by the total of task execution time and the span it needs to write the essential n_{out} output files to the storage and retrieve n_{in} as similar from the storage. Regard that, there is no essential to read the file of input when it is previously accessible in the VM where the task will run. This happens whenever parent and child tasks execute on the same VM.

$$TPT_t^{vmt} = ET_t^{vmt} + \left(\sum_{i=1}^{n_{input}} N_{di,vmt}^{in}\right) + \left(\sum_{i=1}^{n_{output}} N_{di,vmt}^{in}\right) \tag{4}$$

The using resource cost r_{vmt} of category vmt for $lease_r$ units of time is described as

$$C_{rvmt} = \lceil (prov_{vmt} + lease_r)/\tau \rceil \times c_{vmt} \tag{5}$$

Lastly, we believe the data transfers in/out from/to general storage approach are charged free, as is the instance for commodity like Amazon S3, Rackspace Block Storage and Google Cloud Storage. Because of the general storage system, many cloud vendors charge by the found on the stored data amount. We do not add this cost in the total cost estimation of neither our implementation and nor the implementation of other approaches used in the experiment for comparison. The explanation for this having the ability to compare our strategy with others planned to transfer data's in peer-to-peer fashion. Moreover, despite, the stored data amount for a considered workflow is most probably similar in every situation that it does not makes in a variation in cost.

5 The SRPSM algorithm

We present SRPSM, a dynamic algorithm based on heuristic that generates scheduling and resource provisioning selection to fulfill each individual workflow deadline while reducing the price of renting VMs. Its clarity was an important plan objective to improve its execution in real word WaaS platforms and to guarantee its scalability in terms of the number of tasks and workflows. Generally, the algorithm handles a cache of resources which is scaled out/in found on the present requirements of tasks that are start for processing. Its important objective is to competently use these resources as a cost-effective manner without violate the deadlines of each individual workflow. An outline of SRPSM scheduling scenario is depicted in Fig. 4 and a complete heuristic is as follows.

Once a workflow is presented to the workflow scheduler, it is pre-processed and portion of deadline is allocated to each task. This portion of deadline will lead the selections made at processing time when planning each task on to either a new provisioned or an existing resource. The first stage is the strategy of deadline distribution is to estimate the earliest finish time for each task in a workflow described as $eftt = max_{p\epsilon t}.parents\{eftp\} + TPT_t^{VMT}$, where *vmt* related to the category of VM with the high portion of CPU capacity. For clarity, from this moment, we will mention to this category of VM as the fastest category and to the category of VM with the less portion of CPU capacity as slowest category. In this manner, estimated the task processing time utilizing *vmt* guides to the highest value (slowest processing time) but possibly the lesser cost (believing the cost is proportional to the capacity of CPU).

Then, the workflow makespan (i.e. total execution time), is described as $m_w = max_{t\epsilon T}\{eft_t\}$, is estimated. If this value run over the workflow's limit of deadline, then recalculated the earliest finish time of tasks using the next fast processing category of VM until the makespan value is equal or less than the value of deadline. We believe the value of deadline is every time adequate and therefore do no examine instances in which the fast processing accessible VM category still guides to a makespan that is exceed the workflow's deadline. An opportunity for WaaS vendors in this situation should be to dismiss the workflow execution or renegotiate the requirements of QoS with users.

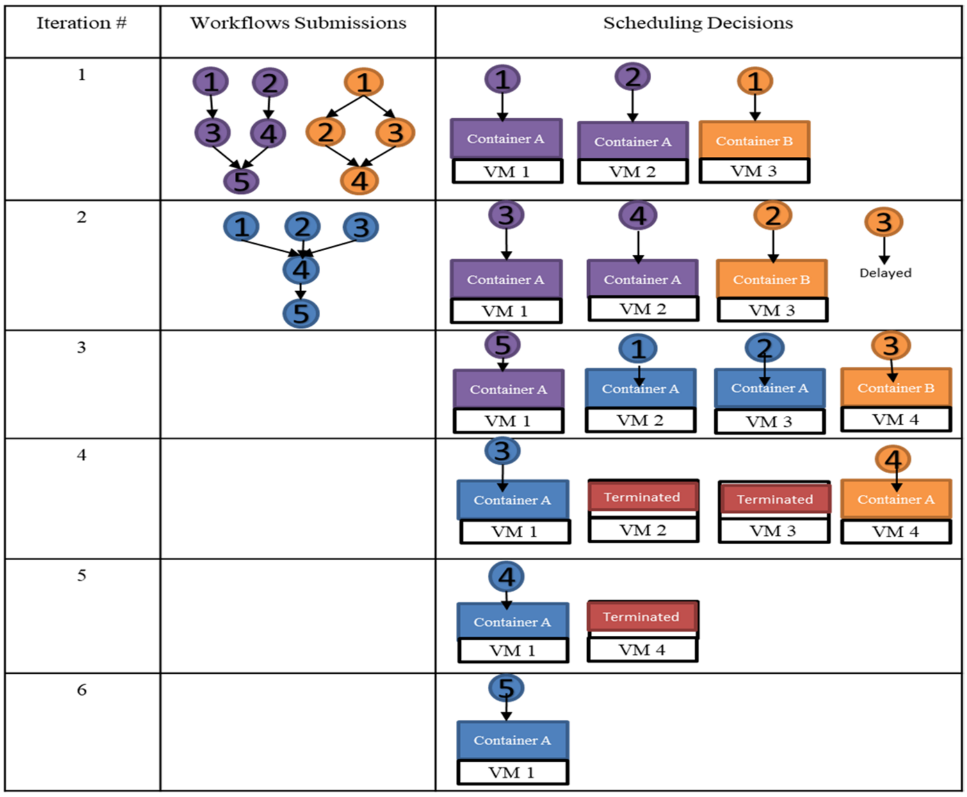


Fig. 4 The SRPSM resource provisioning and scheduling strategy

ALGORITHM 1
SRPSM Scheduling

```
procedure sCHEDULEQUEUEDTASKS(q)
    sort q by ascending deadline (EDF)
    while Q is not empty do
        $t = q.peek$
        $dag = t.dag$
        $container = t.container$
        $vm = null$
        $VM_{idle}$=group of all idle VMs
        $VM_{idle}^{in}$=group of vm $\epsilon$ $vm_{idle}$ that have t's input data
        $vm= vm \epsilon vm_{idle}^{in}$ that can complete t on time with less cost
        if $vm ==null$ then
            $VM_{idle} = VM_{idle} \backslash VM_{idle}^{in}$
            $VM_{idle}^{cn}$= group of all VM $\epsilon$ $vm_{idle}$ with container deployed
            $vm = vm \epsilon vm_{idle}^{cn}$ that can complete task on time with less cost
            if $vm ==null$ then
                $VM_{idle} = VM_{idle} \backslash VM_{idle}^{cn}$
                $vm = vm \epsilon vm_{idle}$ that can complete task on time with less cost
            end if
        end if
        if $vm\ !=null$ then
            if $vm.cn\ !=cn$ then
                $deployCn\ (vm, cn)$
            end if
            q.poll
            scheduleTask(t, vm)
        end if
    end while
end procedure
```

After acquiring an appropriate makespan, the portion of spare time accessible described as the variation between the deadline and makespan (i.e. $dl_w - m_w$) is estimated. This time of spare is then allocated to each individual task in a manner that is proportional to their processing time, that is, tasks with higher processing time get allocated a higher amount of the spare time examine the performance to tasks with lesser processing time. Lastly, each task is allocated a deadline $dlt = t \cdot start\ time + TPT_t^{VMT} + t \cdot spare\ time$.

When a workflow DAG is preprocessed, then scheduling of workflow task can start, this procedure is shown in Algorithm 1. Its major aim is to refuse renting new VMs when achievable rather than re-use existing VMs. In this manner, the consequence of newly provisioning VM delays in respect of uncertainty and cost are minimized and the resources are utilized effectively. This extremely guides to less number of VMs utilized and consumed less billing slots.

Initially, all the arrival tasks (those that possess no parent) in the workflow turn start for execution and are put in a queue for scheduling. As the progresses of workflow execution and tasks are finished, child tasks that are start to execute (those that possess parents and have completed their execution) are dispatched on to the queue. Therefore, at any considerable moment, this queue includes whole tasks from whole workflows are presented to the framework that are start to be scheduled.

Every scheduling interval, which happens every Sch_{int}, every task in the queue is executed in the subsequent manner. The first stage is to identify a VM in idle that can complete the task within the time with lesser cost and map the task on that VM. When calculating the time required to run a task on an existing VM, not only consider the TPT_t^{VMT}, but also consider the container provisioning delay $prov_{cn}$ in situations in which the container need to run the task demands to be used on the VM.

The idle VM is initially searched for in the group VM_{idle}^{in} which is comprised of all presently idle VMs that include part or complete task's input data. In this manner, parent and child tasks are regularly supported to execute on the same VM. The explanation for this are to minimize the utilization of the networks in data center because they are familiarly studied bottlenecks and origin of uncertainty, and therefore to minimize the total processing time of tasks because the input data does not demand to be transported from the general storage approach and consequently to minimize cost by incurring in shorter billing slot. Moreover, by examining container provisioning delays when calculating the cost and processing time of tasks on rented VMs, an idle VM with the matching container used on it will regularly be recommended if it does not guide to the deadline violation.

If no appropriate VM is discovered in VM_{idle}^{in}, then the algorithm attempts to re-use VM from the group VM_{idle}^{cn} including all the idle VMs in which the container linked to the task's workflow is deployed currently. In this manner the provisioning delay of container $prov_{cn}$ is removed. If the group does not include a VM that can complete the task within the time, then the algorithm focus for any existing VM remaining in idle that can fulfill the deadline with lesser cost.

If a suitable existing VM is discovered, then the task is quickly mapped on it. If not found, then as a final call on to re-use a rented VM, the task is wait longer to be mapped in a following scheduling interval, but only if does not guide to a violation of deadline. Therefore, the option to wait longer a task is made formed on the task's processing time on the slowest category and the remaining amount of time to finish the task within the time. In particular, if mapping the task on a next scheduling interval on the slowest category of accessible VM still guides to the task completing the task by its deadline, then the task is wait longer so than it can be possibly mapped on an existing VM on a following interval.

If the task cannot be wait longer, then the category of VM that can complete the task on time with lesser cost is selected. If there is no such category of VM exist then the fastest category of VM that can complete the task is selected. When calculating the cost and processing time of tasks on various VM categories, our approach examines the task's execution time, provisioning delay of VM and initialization delay of container. A VM of the chosen category is then offered, the matching container is deployed, and scheduled the task on it.

To well adapt to environmental unpredictability's and unpredicted delays, all the time a task completes either later than usual or earlier, the deadline of the leftover tasks in workflow is improvised. In this manner, if a tasks completes its execution earlier, child tasks will possess extra time to execute and therefore they would either be allocated to a lesser cost VM or wait longer to be mapped in following intervals. If a task complete its execution later than predicted, modifying the deadline of the leftover tasks should avoid the deadline violation.

Considering the strategy of provisioning of resources, as stated before, new VMs are just provisioned if tasks cannot be wait longer any more. On the contrary of provisioning is deprovisioning, it is a non-negligible one for terminate the idle container (for required workflow container deployment) and idle VM (save some cost and increase the utilization) during the execution. As for the procedure of deprovisioning, monitored every leased VMs at every $prov_{int}$. Hence, first it checks the group of idle VMs in which it first looked the idle VMs for in the group of VM_{idle}^{cn}, containing the VMs container idle and check any VMs' container idle time is equal or exceed the fixed average idle time (it fixes depends on the workload of workflow) if found, container will be terminated in the VM. Then, that VM belong to the group of VM_{idle}^{in}. So that VM can use if any workflow task required to execute with its corresponding container provisioning delay and prevent the newly VM provision to save cost, time, and increase the utilization.

Next it looked the idle VM for in the group of VM_{idle}^{in}, including all the idle VMs and check any idle VM is reached either equal or above the fixed average time (it fixed depends on the workload of workflow), then that VM will be terminated to save some cost, time and increase the utilization. This deprovisioning strategy is depicted in Algorithm 2. It is useful referring that both strategy of scheduling and provisioning intervals (*sch_{int} and $prov_{int}$*) are customizable factors that can be given as input to the algorithm and their weights guide to the balance between performance with regard to makespan (overall execution time) and cost running time used on scheduling and provisioning procedures.

ALGORITHM 2
Resource Deprovisioning

```
procedure mANAGERESOURCES
    $VM^{idle}$ = all leased VMs that are currently idle
    for each $vm_{idle}$ in $VM_{idle}^{cn}$ do
        if $VM_{idle} = VM_{idle} \setminus VM_{idle}^{cn}$ then
            $t_d$ = deprovisioningDelayEstimate for container
            if $t_d >= average$ then
                terminate $cn_{idle}$
            end if
        end if
    end for
        for each $vm_{idle}$ in $VM_{idle}^{in}$ do
        if $VM_{idle} = VM_{idle} \setminus VM_{idle}^{in}$ then
            $t_d$ = deprovisioningDelayEstimate for $vm$
            if $t_d >= average$ then
                terminate $vm_{idle}$
            end if
        end if
    end for
end procedure
```

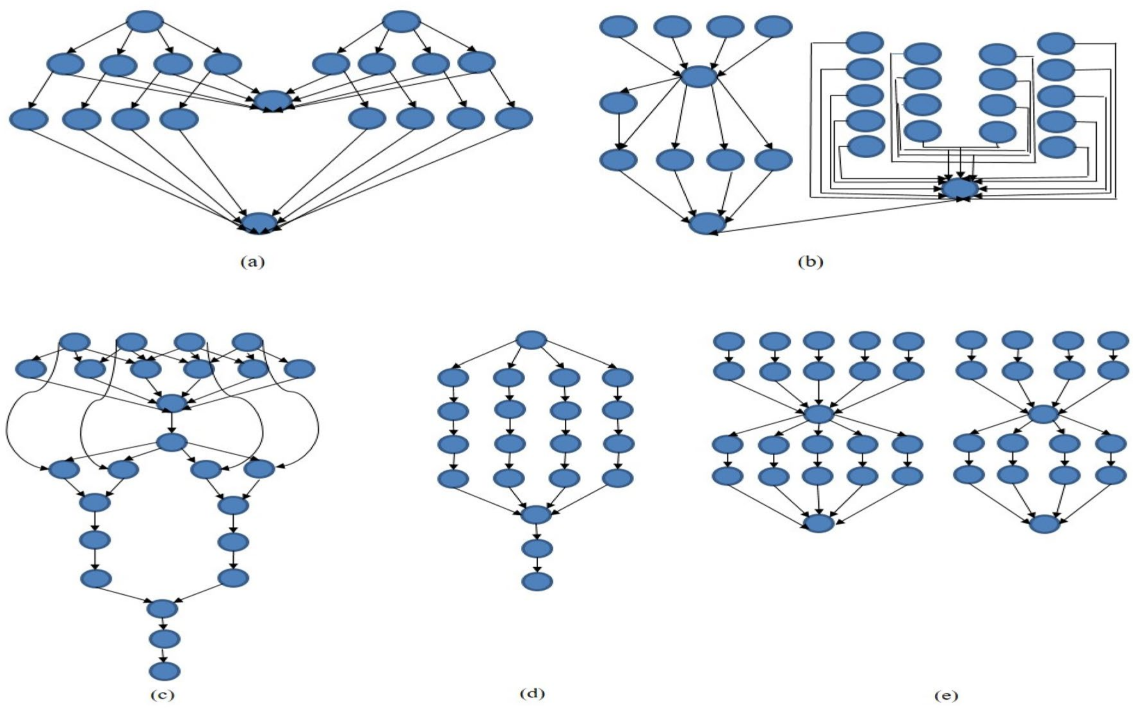


Fig. 5 Five different scientific workflows **a** Cybershake, **b** SIPHT, **c** Montage, **d** Epigenomics and **e** LIGO

Table 1 Categories of VMs used in the evaluation

Name	Capacity of CPI (MIPS)	Price per minute
Extra-large	16	$0.266
Large	8	$0.066
Medium	4	$0.033
Small	2	$0.016

6 Evaluation of performance

Our proposal performance was evaluated using familiar workflows from five various scientific domains. The field of astronomy related application of Montage is used to make big output image of the sky found on a batch of input image. The majority of the tasks are distinguished by being I/O intensive as not demanding the processing capacity of CPU plentiful. The workflow of LIGO from the domain of astrophysics is applied to observe gravitational waves. It is comprised mainly of CPU-intensive tasks with huge memory needs. A SIPHT application is utilized in the field of bio-informatics to automate the look for encoding-genes of sRNA. The majority of the workflow tasks required to have much CPU and less I/O utilization. In addition to the domain of bioinformatics, the Epigenomics workflow application is CPU-intensive that automate the different genome-sequencing operation execution. Lastly, the CyberShake is utilized to distinguish the hazards of earthquake by making synthetic seismograms and can be distinguished as a data-intensive tasks with huge memory and CPU demands. These workflow structures are shown in Fig. 5 and their characterization and their full description is done by Juve et al. (2013).

The evaluation was done with different workload including an integration of all the workflows aforementioned of four different sizes: large (1000 tasks), medium (100 tasks), small (50 tasks) and extra-small (30 tasks). The bundle of workload is comprised of a various number of workflows varying from 10 to 20 workflows and various arrival rate which were used by a Poisson distribution. For the observations given here, we used the processing time made for each task as the task size in millions of instructions (MI).

Each workload of workflow was allocated a deadline. To perform this, initially maximum and minimum makespan values were found for each integration of size and type of workflow. The makespan value of maximum was defined as the execution time occurring from processing all tasks sequentially on a slowest type of single VM. The makespan value of minimum was calculated by each task execution time on a fastest type of single one. A deadline among these maximum and minimum values was selected randomly formed on a uniform distribution.

We used the CloudSim (Calheiros et al. 2011) to encourage the execution of containers and workflows. An Iaas vendor presenting a single data region and four categories of VMs was offered. The used VM category configurations are shown in Table 1. Their price and CPU capacity are a version of the optimized compute instance categories (c4) offered by EC2 Amazon and these VMs would be obtained from Google cloud provider as well. A billing period of VM was one minute was considered and for all VM categories. Based on the study of Mao and Humphrey (2012), the provisioning delay of VM was fix to 100 s. Formed on an average 600 MB container size of image, 500 Mbps bandwidth and 0.4 s initialization and 10 s provisioning delay was done by the study of Piraghaj et al. (2017). Performance variation of CPU was modeled after the information findings by Schad et al. (2010) and some other studies (Iosup et al. 2011; Jackson et al. 2010; Maddikunta et al. 2020; Reddy et al. 2014; Priya et al. 2020; Rodriguez and Buyya 2017a, b; Ostermann et al. 2009). The VM performance was diminished by 24% formed on a normal distribution with mean and standard deviation of 10% and 12% respectively. Based on the latter mentioned work, the bandwidth accessible in data center for each transfer of data was degraded by 19% with mean 9.5% and standard deviation 5% based on a normal distribution.

6.1 Performance of algorithm

We differentiate SRPSM with Dyna (Zhou et al. 2015), an approach designed for a same scenario of application. Both outputs vary anyway, in two characteristics. Dyna was designed to provide probabilistic deadline promises and to utilize not only VMs priced statically (e.g. on-demand instances of Amazon), but also utilize dynamically priced VMs (e.g. spot instances of Amazon). Having simple adjustment, we used Dyna to examine non probabilistic deadlines and utilize static VMs specifically. Dyna executes an A star search to produce advance configuration scheme for each task connecting it to a VM category. At processing time, this configuration scheme, as well as instance reuse and consolidation heuristics are used to task schedule. The correspondents of Dyna do not define a sharing policy of VM among workflows and create no make use of containers but instead map tasks on VMs directly. We executed the Dyna with two different versions, first in which any task from any workflow would use any VM without container (mentioned to as Dyna), and next in which only the VMs reused between same type of workflows (mentioned as Dyna-WS).

To show the advantages of sharing VMs and using containers, we executed one extra version of our approach, SRPSM-WCWS. SRPSM-WCWS without the use of containers but believes that VMs can be re-used between tasks associating to the same type of workflow (i.e. between a Montage workflow tasks with 50, and a Montage workflow with 1000 tasks), it is same to Dyna-WS. Lastly, the intervals of provisioning and scheduling for all SRPSM versions were fixed to 1 s and 10 s respectively.

The aim of this group of experiments is to estimate the algorithm performance in respect of cost and having the ability of meet deadlines. We evaluate SRPSM and Dyna also their variants within three different types of workloads made up of 10, 15 and 20 workflows are shown in Table 2. The three bundles of workload arrival rate was fixed to 5 workflows per minute.

Figure 6a–c demonstrate the cost acquired for each of the workloads (small, medium and large) and algorithms respectively. SRPSM acquires the lesser cost in all three workloads and it also demonstrates that extra cost of utilizing containers is very marginal. As predicted, the obtained cost by SRPSM-WCWS is larger than that acquired for SRPSM for the small, medium and large workloads. The explanation for this is that, by utilizing containers and having the ability to reuse any VMs for any workflow task, SRPSM is suitable to utilize the leased VMs efficiently. SRPSM-WCWS on the other side, even though it does not experience in extra costs occurring from utilizing containers, is limited to use tasks exclusively on those VMs only allocated to workflows of the similar type. This additionally demonstrate the advantage of utilizing containers to minimize cost in WaaS platforms. The similar explanation uses for Dyna and Dyna-WS.

When differentiated to Dyna, SRPSM succeeds significantly lesser costs for all three workloads even in spite of the information that Dyna is not influenced by provisioning of container delays. This is because of SRPSM succeeding an efficient utilization of the leased VMs by minimizing the quota of idle time slots.

The workflows percentage that completed within their deadline for each workload and each algorithm are depicted in Fig. 7. All evaluated algorithms have a better performance in this field with all being over 80%. This one is the important benefit of dynamic algorithms because they are having the ability to recover from unpredicted delays due to environmental uncertainties or performance degradation. The little bit variation between the SRPSM and it variant in performance may be cause by two factors. The fact of the first one is that the sharing policy of VM has an effect on the number of leased VM, their categories and how they are reused. The second fact is the fluctuation of statistical outcome from the stochastic characteristic of the simulated process of performance variation. As for the variation in performance among different sort of workloads, this can be defined by the characteristics of the workflows (size, type and deadline) in the workloads in addition to their submission time. Overall, SRPSM, and SRPSM-WCWS outperforms Dyna and it variant.

To study the obtained makespan in those situations in which the deadline was violated. We marked the makespan average to ratio of deadline for each workflow execution samples in which the deadline was violated. The results are depicted in Fig. 8, where the value of ratio larger than one represents a makespan greater than the deadline. The violated deadlines connected to the workflow, SIPHT and its values of ratio are below 1.025 in every situation. This represents that the variation between makespan and deadline was small. Moreover, after studying the workflow samples for which Dyna not succeeded to satisfy the constraint for the SIPHT and Epigenomics workflows the same number of times. This may represent that in those particular situations, the deadline might have been too tight for the algorithms that can be complete within the time. Another feature supporting to this is the information that SIPHT and Epigenomics are CPU-bound (Juve et al. 2010) and therefore are further negatively affected by the performance degradation of VM CPU.

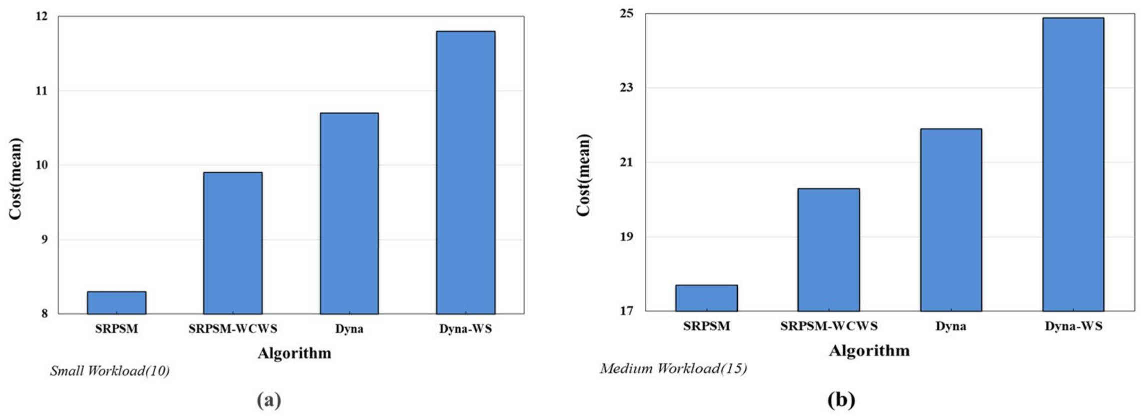
Table 2 Workloads used in the evaluation of the algorithm's performance

Workload name	Number of workflows	Number of tasks
Small	10	500
Medium	15	900
Large	20	5900

Also SRPSM was evaluated with workflows in various arrival rates. For this aim, we managed experiments with five various arrival rates and five workload of 20 workflows. The results are depicted in Fig. 9. The number of VMs and utilization considerably depend on the workload because they are impacted by the possibilities discovered by SRPSM to re-use already leased VMs and it increases the utilization consistently. This represents the algorithm identifies possibilities successfully to better utilize the quota of idle time periods. For the arrival rate of 5 workflows per minutes, anyway, there are large number of tasks at any considerable particular moment in the queue and considerably more number of VMs demand to be rented with the aim of execute them on time, guiding to a less utilization rate and incur high cost. We consider the obtained results for Dyna as a purpose for comparison and in every situation SRPSM outperforms it.

6.2 Sensitivity of provisioning delay

Because of the more number of tasks in the scheduling queue at any considerable particular moment, recurrent VM



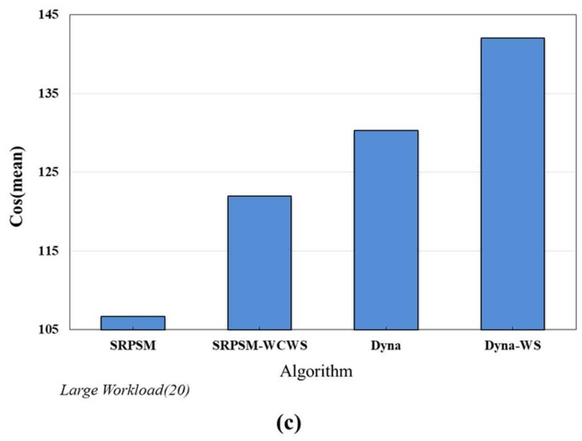


Fig. 6 The executing cost of **a** small (10), **b** medium (15) and **c** large (20) workloads

provisioning procedures may be executed with the aim of satisfy the requirements of deadline of tasks. Hence, it is significant to evaluate the capability of SRPSM to complete the submitted workflows execution with a makespan no larger than the considered deadline within various provisioning delays of VM. Therefore, we evaluated SRPSM within five different provisioning delays of VM varying from 0 to 200 s and the workload belonging to 20 workflows and various rates of arrival. The obtained results for SRPSM and Dyna are shown in Fig. 10.

For SRPSM, frequently use the already leased VM to prevent the new lease VMs when provisioning delay increases and maximize the VM utilization much larger. It is useful observing in addition to the utilization and total number of VMs are not the only characteristic to affect the cost but also they VMs type and how long they are utilized for.

The obtained results for Dyna were added as a reference one and in every situation, SRPSM do better than Dyna on every evaluation metric. Moreover, Dyna lease more VMs to incurred high cost than SRPSM.

Respecting the delays of container provisioning, we evaluated the SRPSM performance with values limiting from 0 to 50 s. The costs are shown in Fig. 11. As predicted, the higher the provisioning delay, the larger the cost as a higher part of a VM's rent time is used initializing containers. Anyway, by admitting the delay of container provisioning when calculating processing time and choosing idle VMs to task schedule, SRPSM attempts to minimize such a cost increase.

6.3 Sensitivity of performance degradation

Knowing variability of performance is significant for schedulers so they can improve from unpredicted delays and satisfy the requirements of QoS. The algorithm sensitivity to CPU degradation performance of VM was done by studying

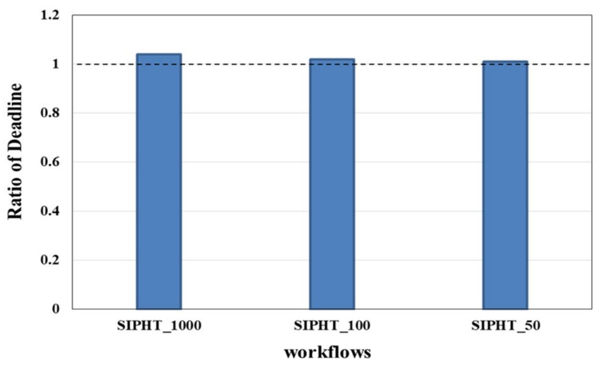
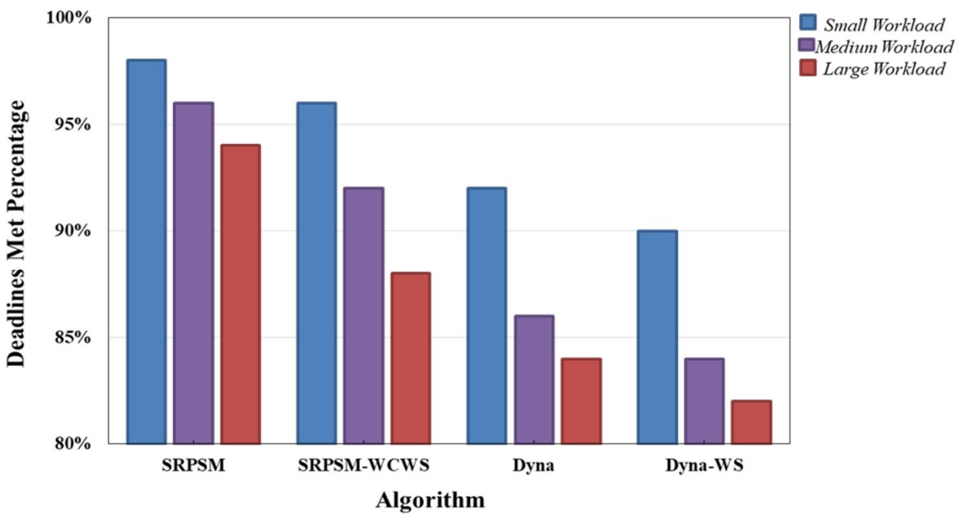


Fig. 8 Average deadline ratio of workflow execution that finished after their deadline for large workload

the deadlines met percentage, VMs utilization average and cost within different values of degradation. The degradation was used using a distribution of normal with variance of 1% and various maximum and average values. The average values were described as half part of the maximum degradation performance of CPU which limited from 0 to 50%.

The obtained results of 20 workflows for workload are shown in Fig. 12. For the workload and algorithms, the deadline met percentage decreases as the percentage of degradation increases, anyway, even with degradation performance maximum of 50%, the deadlines percentage met stays over 80%. It is impossible to fully remove the negative effect of degradation of performance when scheduling the workflows despite if algorithms are adjustable, they still depend on task's runtime estimation to choose decisions. In this manner, even only one task taking longer than usual time may make the deadline to turn inadequate either as its delay impacted its child task making a domino cause, or as the delay was notable sufficient, as it was the final

Fig. 7 The deadline met percentage of small (10), medium (15) and large (20) workloads



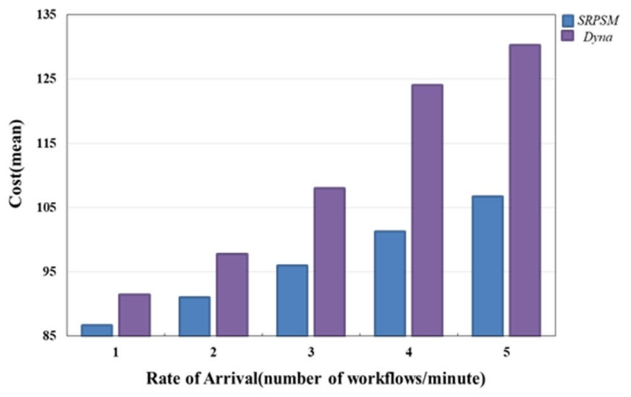


Fig. 9 Cost of executing five workloads of 20 workflows with different arrival rates

workflow task. Especially, we find two features of SRPSM that influence its adaptability. The first feature is the strategy of deadline distribution, which even though repeated all over the workflow execution, is formed on task's run time estimation. The second feature is determination to wait longer the tasks with the aim of support rented VMs, which permits the algorithm to effectively reduce the workflows cost but impacts its responsiveness to improve in the platform. These results shows anyway, that in spite of this, SRPSM is even now successful in reaching its goal of deadline in the vast majority situations.

7 Conclusions and future work

Emerging WaaS platforms with the plan of offering scientists with the potential to utilize their workflow applications for implementation in the cloud in an easy and cost efficient method. Moreover, it can be utilized by multiple users with less cost and also gain more profit to the provider side compared than single workflow execution environment. They possess the ability to transform the manner in which workflows are executed by providing a utility based resource that can be approached on-demand basis from anywhere and by anyone. A significant characteristic, as is for multitenant cloud platform, is to competently handle the execution of multiple workflows associating to various users and with various requirements of QoS. This requires having a scheduling algorithm in scalable in order to efficiently making resource selection for huge number of tasks effectively in addition to a resource provisioning heuristic having the ability of managing the large number of elastic and heterogeneous cloud resources. Therefore, we proposed SRPSM, a dynamic approach developed to schedule various types of workflows in WaaS platforms. Its execution is studied in detail and examined with Dyna, shows not just that SRPSM is having the ability to making high-standard schedules but also the advantages of resource sharing between different workflows in respect of cost which can be done effectively by utilizing containers. Moreover, the arrival rate of 5 workflows per minute can be expandable up to 50 workflows per minute for heavy workloads, just in case need in future (i.e. 100, 500 and 1000 workflows) without modifying the SRPSM procedure.

This assignment is focus on scheduling multiple workflows using containers in WaaS frameworks. There are different features that can be examined to enhance SRPSM and are remain as future assignment. For instance, examining the situation in which images of container are stored on a VM's local storage; this should minimize the use of the data transfers' and possess a significant effect on makespan and cost. For this aim, the available storage amount would be contained as portion of the description of the policies and VM to determine the total number of stored images, their lifespan, and balance between saving I/O data's versus

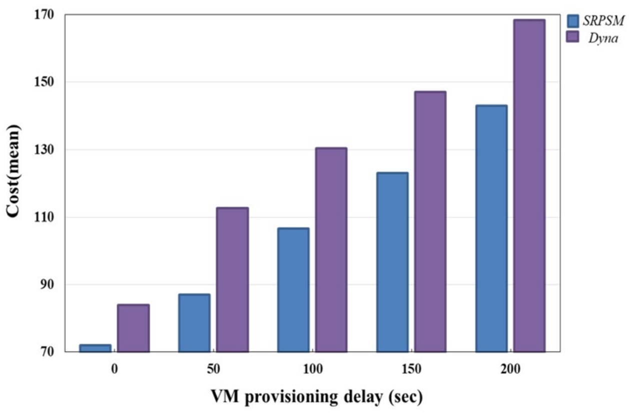


Fig. 10 The large workload (20 workflows) execution cost under different provisioning delay of VM

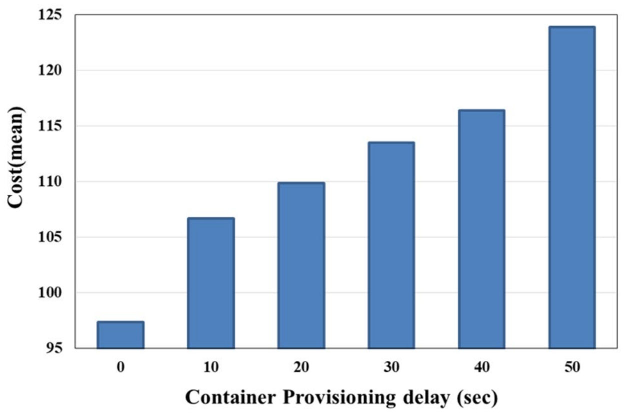


Fig. 11 The large workload (20 workflows) execution cost under different provisioning delay of container

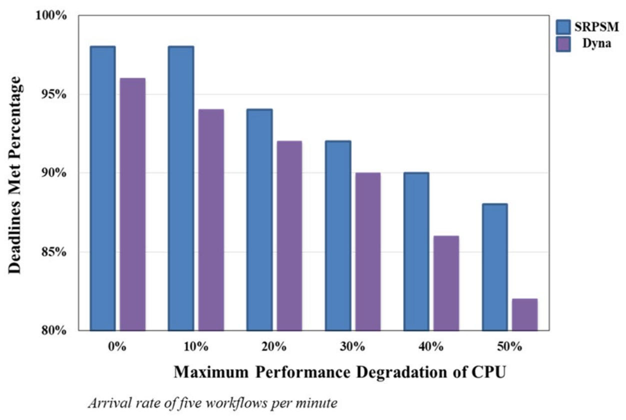


Fig. 12 Deadline met percentage for a workload of 20 workflows with arrival rate and performance degradation of CPU

images would be examined. Another future assignment is to search the usage of multiple containers simultaneously on a single VM with the aim of run multiple tasks in concurrent. Analyzing the impact of resource sharing between different workflows and utilizing containers on the energy consumption is also remain as future assignment. Lastly, it should be of regard for WaaS frameworks in usual to collect and use the execution of workflow data to improve the estimation of tasks' runtimes, to deal privacy and security problems that emerge from their multitenant feature, to design fault tolerant heuristics at different levels of the platform.

References

Abrishami S, Naghibzadeh M, Epema DH (2013) Deadline-constrained workflow scheduling algorithms for infrastructure as a service clouds. Future Gener Comput Syst 29(1):158–169

Bryk P, Malawski M, Juve G, Deelman E (2016) Storage-aware algorithms for scheduling of workflow ensembles in clouds. J Grid Comput 14(2):359–378

Calheiros RN, Ranjan R, Beloglazov A, De Rose CA, Buyya R (2011) CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw Pract Exp 41(1):23–50

Calheiros RN, Buyya R (2013) Meeting deadlines of scientific workflows in public clouds with tasks replication. IEEE Trans Parallel Distrib Syst 25(7):1787–1796

Chai X (2020) Task scheduling based on swarm intelligence algorithms in high performance computing environment. J Ambient Intell Human Comput. https://doi.org/10.1007/s12652-020-01994-0

Chen W, Lee YC, Fekete A, Zomaya AY (2015) Adaptive multiple-workflow scheduling with task rearrangement. J Supercomput 71(4):1297–1317

Chhabra A, Singh G, Kahlon KS (2020) Performance-aware energy-efficient parallel job scheduling in HPC grid using nature-inspired hybrid meta-heuristics. J Ambient Intell Human Comput. https://doi.org/10.1007/s12652-020-02255-w

Deelman E, Singh G, Livny M, Berriman B, Good J, (2008) The cost of doing science on the cloud: the montage example. In: SC'08: proceedings of the 2008 ACM/IEEE conference on supercomputing, IEEE, pp 1–12

Deldari A, Naghibzadeh M, Abrishami S (2017) CCA: a deadline-constrained workflow scheduling algorithm for multicore resources on the cloud. J Supercomput 73(2):756–781

Di Tommaso P, Palumbo E, Chatzou M, Prieto P, Heuer ML, Notredame C (2015) The impact of Docker containers on the performance of genomic pipelines. PeerJ 3:e1273

Dziok T, Figiela K, Malawski M (2016) Adaptive multi-level workflow scheduling with uncertain task estimates. Parallel processing and applied mathematics. Springer, Cham, pp 90–100

Esteves S, Veiga L (2016) WaaS: workflow-as-a-service for the cloud with scheduling of continuous and data-intensive workflows. Comput J 59(3):371–383

Felter W, Ferreira A, Rajamony R, Rubio J (2015) An updated performance comparison of virtual machines and linux containers. In: 2015 IEEE international symposium on performance analysis of systems and software (ISPASS), IEEE, pp 171–172

Filgueira R, Da Silva RF, Krause A, Deelman E, Atkinson M (2016) Asterism: Pegasus and dispel4py hybrid workflows for data-intensive science. In: 2016 Seventh international workshop on data-intensive computing in the clouds (DataCloud), IEEE, pp 1–8

Gerlach W, Tang W, Keegan K, Harrison T, Wilke A, Bischof J, DSouza M, Devoid S, Murphy-Olson D, Desai N, Meyer F, (2014) Skyport-container-based execution environment management for multi-cloud scientific workflows. In: 2014 5th International workshop on data-intensive computing in the clouds, IEEE, pp 25–32

Gil Y, Deelman E, Ellisman M, Fahringer T, Fox G, Gannon D, Goble C, Livny M, Moreau L, Myers J (2007) Examining the challenges of scientific workflows. Computer 40(12):24–32

Gupta A, Bhadauria HS, Singh A (2020) Load balancing based hyper heuristic algorithm for cloud task scheduling. J Ambient Intell Human Comput. https://doi.org/10.1007/s12652-020-02127-3

Iosup A, Ostermann S, Yigitbasi MN, Prodan R, Fahringer T, Epema D (2011) Performance analysis of cloud computing services for many-tasks scientific computing. IEEE Trans Parallel Distrib Syst 22(6):931–945

Jackson KR, Ramakrishnan L, Muriki K, Canon S, Cholia S, Shalf J, Wasserman HJ, Wright NJ (2010) Performance analysis of high performance computing applications on the amazon web services cloud. In: 2010 IEEE second international conference on cloud computing technology and science, IEEE, pp 159–168

Jiang HJ, Huang KC, Chang HY, Gu DS, Shih PJ (2011) Scheduling concurrent workflows in HPC cloud through exploiting schedule gaps. In: International conference on algorithms and architectures for parallel processing, Springer, Berlin, pp 282–293

Jiang Q, Lee YC, Zomaya AY (2015) Executing large scale scientific workflow ensembles in public clouds. In: 2015 44th International conference on parallel processing, IEEE, pp 520–529

Jin H, Wang X, Wu S, Di S, Shi X (2014) Towards optimized fine-grained pricing of IaaS cloud platform. IEEE Trans Cloud Comput 3(4):436–448

Juve G, Deelman E, Vahi K, Mehta G, Berriman B, Berman BP, Maechling P (2010) Data sharing options for scientific workflows on amazon ec2. In: SC'10: Proceedings of the 2010 ACM/IEEE international conference for high performance computing, networking, storage and analysis, IEEE, pp 1–9

Juve G, Chervenak A, Deelman E, Bharathi S, Mehta G, Vahi K (2013) Characterizing and profiling scientific workflows. Future Gener Comput Syst 29(3):682–692

Kouki Y, Ledoux T (2013) RightCapacity: SLA-driven cross-layer cloud elasticity management. Int J Next Gener Comput 4(3):250–262

Liu S, Ren K, Deng K, Song J (2016) A task backfill based scientific workflow scheduling strategy on cloud platform. In: 2016 Sixth international conference on information science and technology (ICIST), IEEE, pp 105–110

Maddikunta PKR, Gadekallu TR, Kaluri R, Srivastava G, Parizi RM, Khan MS (2020) Green communication in IoT networks using a hybrid optimization algorithm. Comput Commun 159:97–107

Maechling P, Deelman E, Zhao L, Graves R, Mehta G, Gupta N, Mehringer J, Kesselman C, Callaghan S, Okaya D, Francoeur H (2007) SCEC CyberShake workflows—automating probabilistic seismic hazard analysis calculations. Workflows for e-Science. Springer, London, pp 143–163

Malawski M, Juve G, Deelman E, Nabrzyski J (2015) Algorithms for cost-and deadline-constrained provisioning for scientific workflow ensembles in IaaS clouds. Future Gener Comput Syst 48:1–18

Mao M, Humphrey M (2011) Auto-scaling to minimize cost and meet application deadlines in cloud workflows. In: SC'11: proceedings of 2011 international conference for high performance computing, networking, storage and analysis, IEEE, pp 1–12

Mao M, Humphrey M (2012) A performance study on the vm startup time in the cloud. In: 2012 IEEE fifth international conference on cloud computing, IEEE, pp 423–430

Ostermann S, Iosup A, Yigitbasi N, Prodan R, Fahringer T, Epema D (2009) A performance analysis of EC2 cloud computing services for scientific computing. In: International conference on cloud computing, Springer, Berlin, pp 115–131

Pietri I, Malawski M, Juve G, Deelman E, Nabrzyski J, Sakellariou R (2013) Energy-constrained provisioning for scientific workflow ensembles. In: 2013 International conference on cloud and green computing, IEEE, pp 34–41

Piraghaj SF, Dastjerdi AV, Calheiros RN, Buyya R (2017) Container-CloudSim: an environment for modeling and simulation of containers in cloud data centers. Softw Pract Exp 47(4):505–521

Priya RMS, Bhattacharya S, Maddikunta PKR, Somayaji SRK, Lakshmanna K, Kaluri R, Hussien A, Gadekallu TR (2020) Load balancing of energy cloud using wind driven and firefly algorithms in internet of everything. J Parallel Distrib Comput 142:16–26

Rajan CDS (2020) Design and implementation of fuzzy priority deadline job scheduling algorithm in heterogeneous grid computing. J Ambient Intell Human Comput. https://doi.org/10.1007/s12652-020-02171-z

Reddy GT, Sudheer K, Rajesh K, Lakshmanna K (2014) Employing data mining on highly secured private clouds for implementing a security-asa-service framework. J Theor Appl Inf Technol 59(2):317–326

Rodriguez MA, Buyya R (2017a) Budget-driven scheduling of scientific workflows in IaaS clouds with fine-grained billing periods. ACM Trans Auton Adapt Syst 12(2):1–22

Rodriguez MA, Buyya R (2017b) A taxonomy and survey on scheduling algorithms for scientific workflows in IaaS cloud computing environments. Concur Comput Pract Exp 29(8):e4041

Schad J, Dittrich J, Quiané-Ruiz JA (2010) Runtime measurements in the cloud: observing, analyzing, and reducing variance. Proc VLDB Endow 3(1–2):460–471

Shi J, Luo J, Dong F, Zhang J (2014) A budget and deadline aware scientific workflow resource provisioning and scheduling mechanism for cloud. In: Proceedings of the 2014 IEEE 18th international conference on computer supported cooperative work in design (CSCWD), IEEE, pp 672–677

Stavrinides GL, Karatza HD (2015) A cost-effective and qos-aware approach to scheduling real-time workflow applications in paas and saas clouds. In: 2015 3rd international conference on future internet of things and cloud, IEEE, pp 231–239

Thennarasu SR, Selvam M, Srihari K (2020) A new whale optimizer for workflow scheduling in cloud computing environment. J Ambient Intell Human Comput. https://doi.org/10.1007/s12652-020-01678-9

Vöckler JS, Juve G, Deelman E, Rynge M, Berriman B (2011) Experiences using cloud computing for a scientific workflow application. In: Proceedings of the 2nd international workshop on scientific cloud computing, pp 15–24

Wang W, Niu D, Li B, Liang B (2013) Dynamic cloud resource reservation via cloud brokerage. In: 2013 IEEE 33rd international conference on distributed computing systems, IEEE, pp 400–409

Wang J, Korambath P, Altintas I, Davis J, Crawl D (2014) Workflow as a service in the cloud: architecture and scheduling algorithms. Procedia Comput Sci 29:546

Xu M, Cui L, Wang H, Bi Y (2009) A multiple QoS constrained scheduling strategy of multiple workflows for cloud computing. In: 2009 IEEE international symposium on parallel and distributed processing with applications, IEEE, pp 629–634

Yin W, Mavaluru D, Ahmed M, Abbas M, Darvishan A (2020) Application of new multi-objective optimization algorithm for EV scheduling in smart grid through the uncertainties. J Ambient Intell Human Comput 11(5):2071–2103

Yu Z, Shi W (2008) A planner-guided scheduling strategy for multiple workflow applications. In: 2008 International conference on parallel processing-workshops, IEEE, pp 1–8

Zhou AC, He B, Liu C (2015) Monetary cost optimizations for hosting workflow-as-a-service in IaaS clouds. IEEE Trans Cloud Comput 4(1):34–48