Optimal Resource Provisioning for Scientific Workflow Ensembles in Cloud Computing

Abstract. A workflow ensemble consists of multiple sub-workflows, each containing hundreds or thousands of jobs with constrained priorities. Their implementations are widely accepted. Furthermore, extensive research and development work has been done to automate them and make them effective in cloud computing environments. This paper addresses the challenge of resource provisioning for large-scale scientific workflows running on many heterogeneous clouds by running all processes of the ensemble under budget and time constraints. We have developed four algorithms for resource provisioning and job scheduling based on static and dynamic methods. Simulations based on real scientific workflow applications are applied in experimental evaluation. The results show the ability of the proposed approach of reducing budgets and execution times for various workflow ensembles.

Keywords: Workflow ensembles, Resource provisioning, Scheduling algorithms, Cloud computing.

1 Introduction

Scientific Workflows have been widely utilised to model large scale scientific and engineering application in several fields such as Earth Science, Astronomy, Physics, and Bio-informatics [1,2]. They describe a set of computations that enable data analysis in a structured and distributed manner, usually expressed as a set of tasks and a set of dependencies between them. Many applications in science and engineering are increasingly formed as workflows with many precedence-constrained jobs, e.g., Montage [3], LIGO [4], and CyberShake [5]. Scientists are required to execute these workflows with different parameters repeatedly or use a combination of different workflows to achieve an ultimate goal. We use the term workflow ensemble to represent an entire scientific analysis as a set of interrelated but independent workflow applications. A workflow ensemble usually contains many sub-workflows in each of which hundreds or thousands of jobs exist with precedence constraints.

To efficiently execute scientific workflows, scientific workflow management systems (SWMS) typically use high-performance computing (HPC) resources in cluster, grid, or cloud environments. Cloud computing that is considered the latest distributed computing paradigm that provides several advantages for delivering these applications. Infrastructure-as-a-Service (IaaS) clouds, in particular, offers a SWMS. Lease virtualized computing resources or virtual machines (VMs) for easily accessible, flexible, and scalable infrastructure [6]. The resource provisioning phase determines the amount and the type of resources required and requests the resources for workflow execution. Resource allocation is defined as the assignment of reserved resources to tasks in the workflow. The scheduling phase then maps the workflow tasks to the provisioned resources [7]. The multitask workflow scheduling on parallel and distributed systems is extensively studied over the last years. It is perceived NP-Hard Problem [8, 9]. Therefore, it is very difficult to produce an optimal solution within the polynomial time. The utility-based pricing model offered by cloud providers means that finding a trade-off between cost and performance is a common denominator for scheduling algorithms. This is done mostly by trying to minimise the total infrastructure cost while meeting a time constraint, or deadline [10]. There are a good number of studies [10, 11, 13] working on the monetary cost and performance optimizations for scientific workflows. Some heuristics adopted in the previous works, such as the deadline assignment heuristic [12] have demonstrated less effective than the more comprehensive approach to explore the solution space [14]. In this case, saving costs when running the ensemble on the cloud by using a cost-aware scheduler is more appropriate for other purposes. For other purposes, e.g., to save costs when running the ensemble on the cloud, using a cost-aware scheduler is more appropriate in this case [15, 16]. On the other hand, if you need to run faster, it is better to use a makespan (execution time) aware scheduler. The stated objective is to create a schedule that minimizes the makespan of the entire workflow ensemble [17].

This paper proposes a general solution for running workflow ensembles in cloud computing. In particular, we address the cost optimization problem of large-scale scientific workflows running on multiple heterogeneous clouds by executing all workflows from an ensemble under budget and deadline constraints.

Our contributions, in this paper, are as follows:

1. Execute all workflows from an ensemble workflow with respect of priority and deadline.
2. Minimize the number of VMs capable of running all workflows from an ensemble workflow.
3. Find the minimal budget by a selective resource provisioning (reserved, on-demand and spot).
4. Evaluate the proposed model using a simulator based on WorkflowSim [35], which models the infrastructure and the application, taking into account uncertainties in task runtime estimates and provisioning delays.

This work has been divided into 5 sections, each of which deals with a certain issue. Section 2 presents an outline of the related work. We have described the proposed model and assumptions of hybrid cloud environment in Section 3. Section 4 discusses the performance evaluation of the proposed model. Finally, conclusions are stated in Section 5.

2 Related work

Lots of work has addressed the issue of task scheduling and provisioning of resources for workflows for cloud infrastructures [18, 19, 20]. Huu and Montagnat [20] propose cost-based algorithms for workflow-based applications aiming at minimizing the number of allocated resources and reducing application makespan. Other processes provided in [21, 22] take into account unpredictable dynamic workloads on IaaS clouds and optimize objectives like cost, runtime, or utility function by auto-scaling the resource pool at runtime. Deelman et al. [23] studied the cost of running scientific workflows in the cloud. Specifically, they studied the trade-off between cost and performance under different execution and resource provisioning plans as well as storage and networking fees on Amazon AWS. Their findings support the fact that clouds are a cost-effective solution for scientific applications. Our approach is different, since we study workflow ensemble for deadline constraints and minimization cost. When Scheduling multiple workflows in [24, 25, 26], time and cost optimization with multiple goals under memory bandwidth and capacity constraints is done using game theory heuristics. Malawski et al. [27] developed a set of algorithms to intensify the efficiency of executing scientific workflow ensembles on IaaS. Authors design Dynamic Provisioning Dynamic Scheduling (DPDS) an online algorithm, providing schedules and resources at run time. In this work, ensembles are created using the synthetic workflows from the real applications like LIGO, CyberShake etc. Overall, three different algorithms are designed: DPDS, Workflow-Aware DPDS (WA-DPDS) which are dynamic algorithms, and one static algorithm: Static Provisioning Static Scheduling (SPSS). The simulation results demonstrate that SPSS and WA-DPDS (which takes into account the structure of workflows and task runtime) provide better result than the simple priority based scheduling that makes decisions based on resource utilization only. These algorithms are tested using the WorkflowSim simulator. It simulates the infrastructure and application while accounting for task runtime uncertainties. This work seeks to maximize the number of user-prioritized workflows that can be executed within the restrictions of the budget and deadline. Our work is related to the strategies for deadline constrained cost-minimization workflow scheduling. We execute all workflows from an ensemble workflow instead of a subset of the submitted workflows. We minimize the cost by exploiting all instances offered by the provider (reserved, on-demand and spot).In [28], they explore the area of workflow ensemble scheduling algorithms that are aware of the underlying storage architecture and can consider data transfers between tasks when scheduling ensembles. This work is motivated by determining how data transfers influence ensemble execution under budget and deadline constraints. It also answers the following question; how should execution systems handle data-intensive ensembles? They introduce a new scheduling algorithm that takes advantage of file caching to speed up ensemble execution. However, they ignore deadline constrained and cost-minimization. In [29], they offer Deco, a declarative optimization engine that can automatically build resource provisioning plans for scientific workflows in the cloud bearing in mind the dynamicity of cloud performance. With Deco, users only need to declaratively specify their performance/monetary cost goals and constraints and leave the rest to the optimization engine. Moreover, a novel approach of probabilistic optimizations on goals and constraints is developed to address cloud dynamics. In [30], the authors propose a flexible schedule for workflow ensembles in clouds. This allows replacing the objective function according to the user's needs. The proposed schedule uses a method called Particle Swarm Optimization (PSO), a population-based heuristic search algorithm. All these works present various strategies to schedule a set of workflows simultaneously under different constraints. In our work, we execute all the workflows by reserving the minimum number of VMs. Then, we minimise the cost by a selective resource provisioning (reserved, on-demand and spot). The proposed algorithms take into consideration deadline constrained and user's priority.

3 Proposed model

This paper formulates the cloud workflow-scheduling problem in clouds from four fundamental aspects: workflow model, cloud resource model, cost model, and objective model.

3.1 Cloud resource model

A cloud consists of an unlimited number of homogeneous single-core virtual machines (VMs) that can be provisioned and deprovisioned at any time such as Amazon EC2. Input and output data are stored on a cloud object store such as Amazon S3. VMs can execute only one task at a time. A VM not executing any task is called an idle VM. When a task is submitted to a non-idle VM, it is enqueued for execution. When a task successfully finishes execution on a VM, a queued task (if any) is dequeued and executed according to the FIFO rule. In this model, all VM instances are billed by the hour of use, ignoring charges related to data transfer to and from the cloud. Amazon provides on-demand and reserved VM instances that are associated with a fixed set price [31]. Since December 2009, Amazon released a new type of instances called Spot Instance (SI) to sell the idle time of Amazon's EC2data centers [32]. The price of an SI, spot price, depends on the type of instance as well as VM demand within each data center. The users provide a bid that is considered the maximum price to be paid for an hour of usage. Whenever the current price of an SI is equal or less than the user bid, the instance is made available to the user. If the price of an SI becomes higher than the user's bid, the VM(s) will be terminated by Amazon automatically and Moreover, the user does not pay for any partial hour. However, if the user terminates the running VM(s), she has to pay for the full hour. Amazon charges users per hour by the market price of the SI at the time of VM creation [33].

3.2 Workflow model

Normally, a workflow is a set of tasks that will be submitted to the clouds for execution. A workflow application G=(T,E) is modeled as a Directed Acyclic Graph (DAG) where T={t1,t2,…,tn} is the set of tasks and E is the set of directed edges. An edge eij of the form (ti,tj) exists if there is a data or control-flow dependency between ti and tj, case in which ti is said to be the parent task of tj and tj is said to be the child task of ti. Based on this definition, a task can only start after all of its parent tasks have finished and all necessary data has been received [34]. Target applications are ensembles of workflows (sets of workflows). Workflow tasks have runtime estimates that indicate how long it takes to execute them on a given VM type.

3.3 Workflow Scheduling Problem

In IaaS clouds, users can rent computers to do their work and they have to pay for the time they use the computer. The cloud offers different types of instances with various abilities and costs. The problem of workflow scheduling revolves around choosing the best type of computer for each task in order to satisfy users' optimization requirements.

1. We assume that the estimated execution times for the workflow tasks are known.
2. Although workflows are often data-intensive, the algorithms described currently here do not consider the size of the input and output data when scheduling tasks.
3. Each workflow in a set has a numerical priority that indicates how important the workflow is to the user. Therefore, the priorities indicate the user's utility function. These priorities are absolute in the sense that completing a workflow with a given priority is more valuable than completing all other workflows in the set with lower priorities combined.
4. Cloud deployment and workflow set planning problem seeks to complete as many high priority workflows as possible with a fixed budget and deadline. Only workflows where all tasks are completed before the deadline are considered complete. Subtotals cannot be used in this model.

The purpose of our work is to guarantee running all workflows in the deadline specifies with a minimum budget. To achieve this, we have developed scheduling algorithms and simulations in order to achieve the underlined objectives. Initially, we have a set of workflows that we seek to execute in an affordable deadline (the deadline must be greater than the maximum Critical path of these workflows). Then, we must determine a minimum VM number in which execution of all workflows is completed before the deadline. We are also interested in the type of VM; therefore, we take three types of payments which are: VM Spot, VM reserved, and VM on demand. We have proposed two workflow priority approaches: priority is chosen by the user or our proposal to prioritize workflows in the Critical path order (the workflow that has a longer Critical path is given higher priority).

3.4 Algorithms

This section explains the algorithms that were developed to schedule and provision resources for ensembles of workflows on the cloud under budget and deadline constraints. Our proposition underscores on setting priorities for each workflow by user, by critical path and order priorities.We will make the workflow that has greater critical path, highest priority and preference the shortest tasks (if multiple tasks have the same priority are available). This reduces the timeout of waiting and makes the execution finished faster than others.

3.4.1 Provisioning

At the beginning, for each workflow, we determine the list of all its levels and we calculate its critical path. Then, we gathered all the selected workflows to have the task list of all the workflows. Our intended objective is to determine the minimum number of VMs to run all workflows in accordance with the deadline. Therefore, for this, we need to determine the minimum and maximum number of VMs to run all workflows is required.

The initial minimum VM number was defined as follows:

$$MinVm = \frac{\sum_{w \in \text{Workflow}}(\sum_{t \in \text{task}} T_{\text{execution}})}{\text{Deadline}} \quad (1)$$

MinVM is the minimum number of VMs to execute a set of workflows in the specified deadline.**Therefore**, we assumed that the scheduling of the tasks is executed independently (there are no dependencies between the tasks).

And for the maximum number of VMs:

$$MaxVm = \sum_{w \in Workflow}(Max\_Tasks\_number(\text{Level} / \text{L} \in \text{w})) \quad (2)$$

In order to **ascertain** the maximum number of VMs required, we determine the level that contains a maximum number of tasks for each workflow (we give the number of VMs to execute the workflow and we make the sum of result for all workflows, which represents the maximum number of VMs). To make the MaxVM more accurate and decrease the range between the MaxVM and MinVM, we distribute vertically the tasks of the level that have the maximum number of tasks until we reach the deadline (which give us the maximum number of VMs to execute the workflow, and we make the sum of result for all workflows).Thus, we get an adapted relation between the deadline and the MaxVM. We created a function to distribute for each workflow the tasks in the levels according to the deadline called Vertical_Distribute_Task. This function will work only if the deadline is higher than critical path. **It** will try to stretch the levels until reaching or be very closer to the deadline (See Figure 1).

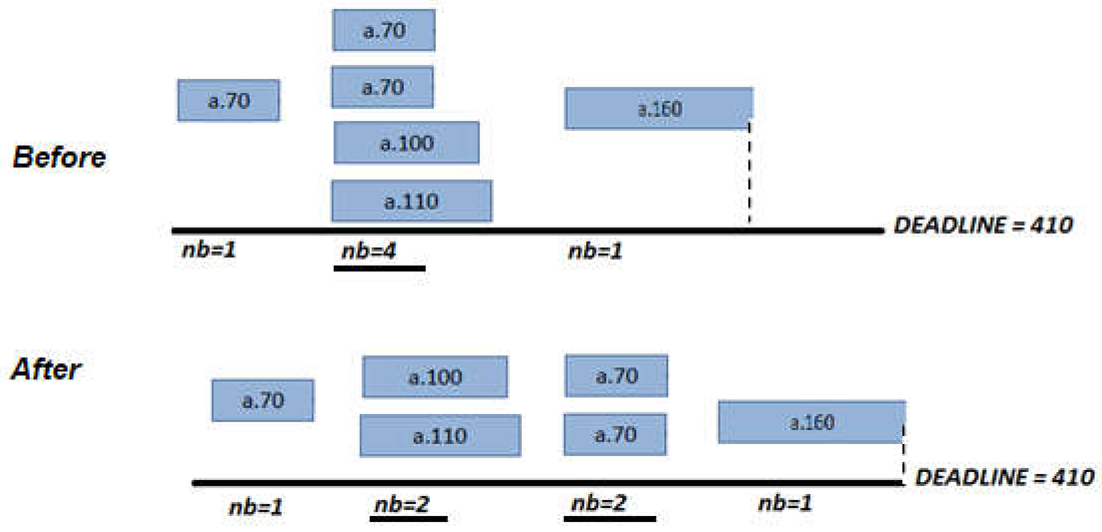


Fig. 1 Example of Vertical Distribute Tasks

While the Deadline is higher than the critical path, it will distribute vertically the tasks of the level that have the max number of tasks in two levels. Each level has half number of tasks. Then add the maximum execution time (Texe) of the new level, until we reach the deadline. Consequently, we will get a smaller number of MaxVM. In this algorithm, the function Vertical_Distribute_Tasks creates a new task distribution after the virtual partition.

```
Procedure PROVISIONING( JobList)
Begin
    Total=0 ;
    For task T in JobListdo
        Total=Total + T.Texe ;
    End For
    VMmin= Round(Total/deadline) ;
    For workflow w in workflows ensemble do
    Add level(w) ;
    Calculate Critical Path(w) ;
    End For
    MaxPath= get MaxCriticalpathTime() ;
    For workflow w in workflows ensemble do
    PositionMax= MaxLevelNbrTasks(w) ;
    a= 1- (MaxPath/deadline) ;
    d= deadline * a ;
    Length= w.critical path + Texe(PositionMax) ;
    While ( length < d ) & ( NbrTask(PositionMax) >1 ) do
      Vertical_Distribute_Tasks(PositionMax) ;
      PositionMax= MaxLevelNbrTasks(w) ;
      Length= Length + Texe(PositionMax);
    End while
  End For
End
```

Fig. 2 Pseudo code of the provisioning scheme

3.4.2 Scheduling

In the beginning of scheduling, the tasks of the first level of workflows are assigned to the *jobSubmitted*. Depending on the number of VMs available, some tasks are running and others are waiting. Then, each task that it entire parent finished the execution, it passed to 'ready' state (ready for execution and wait liberation of resource VMs).

The task passed to execution according to its priority. Any VM that finished execution of any tasks, his child tasks passes to ready state and it is added to *jobSubmitted*list (tasks ready to execute). Furthermore, this VM passes to idle state. If *jobSubmitted* contains many tasks, then the task that has great Priority is attached to VM for execution. In addition, if we have more jobs that have same priorities, the tasks that have higher *Texe* passed first.

Scheduling is completed if all tasks are executed. It will be interrupted if the deadline is reached.

```
Procedure Priority-based scheduling(JobList)
   JobSubmitted = empty waiting queue ordered by (Priority, Texe ) ;
   IdleVM= All VMs ;
BEGIN
   For all root Task t in JobList
      Add(t,jobSubmitted) ;
   End for;
   While ( not deadline ) do
      While (jobSubmitted != 0&idleVM !=0) do
         Vm= POP(idleVM) ;
         Task= POP(jobSubmitted) ;
         Submit(Task,Vm) ;
      End while ;
      Wait for the completion of a task T that executes in vm V;
      Add ( readyChild(T) , jobSubmitted);
      Add (V , idleVM);
   End while ;
END.
```

Fig. 3 Pseudo code of the scheduling scheme

3.4.3 Execution

We computed the initial minimal and maximal VM number for our simulation in algorithm 1, and we will use them in this section.

We are interested in performing a dichotomous search in order to determine the number of final minimal VMs with the appropriate scheduling. At each iteration, a simulation is performed with a number of VM = (VMmin + VMmax) / 2 until a minimum VM without exceeding Deadline.

.

```
Procedure EXECUTION
```

```
Begin
    Simulation(nbrVM) ;
    Min= getMinVM() ;
    Max= getMaxVM() ;
    While ( Min< Max ) do
        Moy = (Min + Max)/2 ;
        Simulation(Moy) ;
        if ( Deadline reached ) then Min= Moy +1 ;
        else Max = Moy ;
                endif
    End while ;
    If ( Deadline ) then
                        Min= Min +1 ;
                            Simulation(Min) ;
    Endif ;
End.
```

Fig. 4 Pseudo code of the execution scheme

This section concludes that the final minimum VM number is obtained with the appropriate scheduling and we proceed to the next step which allows us to calculate the minimum budget required for this set of workflows.

3.4.4 Budget Approach

After having the minimum number of VM to execute the set of selected workflows, we are interested to calculate the necessary budget for this execution. We start by loading the SPOT file that contains a list of prices for 24 hours.

```
Procedure Cost(JobList, ,$VM_{min}$,deadline)
Begin
        Read( spotPricingHistory.txt) ;
        $Cost_{min}=0$;
        $Cost_{Reserved}=0$;
        $Cost_{Ondemand}=0$;
        $Cost_{Spot}=0$;
        For eachVMvm in all VMs
        $Cost_{Reserved}$= ReservedCalc(JobList, VM,deadline);
$Cost_{Ondemand}$= $Cost_{Ondemand}$+ OnDemandeCalc(JobList, VM, deadline);
$Cost_{Spot}$= $Cost_{Spot}$+  SpotCalc(JobList, VM, deadline);
$Cost_{min}$ = $Cost_{min}$+  Min($Cost_{Reserved}$, $Cost_{Ondemand}$,$Cost_{Spot}$)  ;
        End for;
End.
```

Fig. 4 Pseudo code of the budget scheme

For each VM, we calculate the price of the performed tasks for the three cases. If the VM is considered reserved, on-demand, or a spot, then, we choose the minimum price. Finally, we calculate the sum of the minimum prices that provides us with the minimum budget.

4 Performance evaluations

Section four deals with the presentation of the details about the experiment carried out to assess the performance of the proposed scheduling algorithms. We seek to execute all workflows tasks respecting the deadline with minimal number of VMs and minimal budget. We have compared our results with two other scheduling algorithms that are RR (Round Robin), and FCFS (First Come First Serve).

4.1 Simulation

In order to evaluate our proposed algorithms, we have implemented our solutions using the CloudSim simulator. However, since it manages only the independent tasks, we have integrated WorkflowSim [35], which provide the capability to simulate scientific workflows (with tasks dependency scheduling).

4.1 Workflows

The ORPWE's performance is assessed on four different scientific workflows such as Montage [36], LIGO [37], and CyberShake [38]. We used another scientific workflow named Sipht. The Montage scientific workflow [36] is an astronomy application that is used to generate custom mosaics of the sky using some set of input images in the ''Flexible Image Transport System (FITS)'' format. Most of its tasks are characterised as I/O intensives and they do not require much computation power. The LIGO ''Laser Interferometer Gravitational Wave Observatory (LIGO)'' scientific workflow [37] is used to identify the gravitational waves produces by various events in the universe. Its tasks necessitate high computing capacity with large memory. Cybershake is utilized in the ''Southern California Earthquake Center'' to illustrate earthquake hazards in particular region by generating synthetic seismograms, and its task can be categorized as I/O intensive which requires significant memory and CPU requirements [38] Sipht is a workflow used in sRNA identification process to automate the search for sRNA encoding. It is a part of bioinformatics project at Harvard University. Further, each of these scientific workflows has a different structure, and composition with characteristics such as (Pipeline, Data Distribution, Data Aggregation and Data Redistribution) [39]. These are presented in the DAX (Directed Acyclic Graph in XML) format in literature [40]. We have considered three sizes of workflow in our experiments small (approx. 50 tasks), medium (approx. 100 tasks) and large (approx. 800 tasks).

4.2 Workflows ensemble

In addition to the unique scientific workflows and in order to evaluate our algorithms on a set of scientific workflows. We have created three different types of workflows composed of all five scientific workflows used in our paper in different ways. We select randomly 10 of these workflows to generate each ensemble as follows:

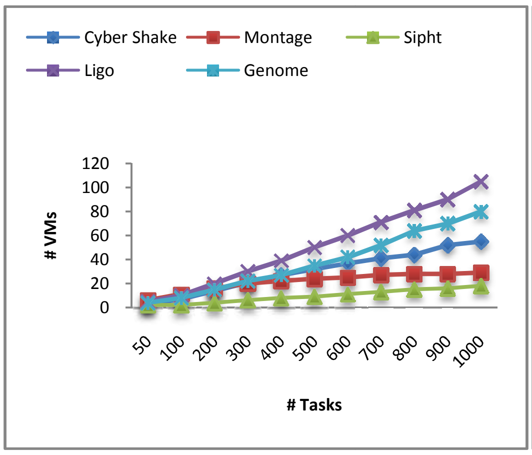
- Constant: contains workflows with the same number of tasks. We have varied the number of tasks from 50 to 500.
- Pareto-sorted workflows contain a small number of larger workflows and a large number of smaller workflows. Workflows with more than 500 tasks are considered to be large workflows.

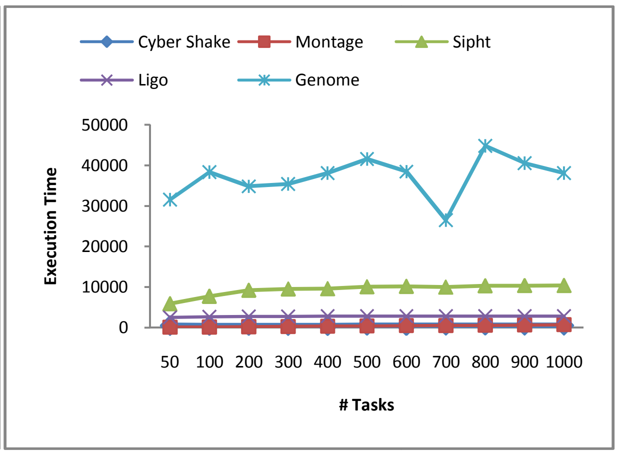
4.3 Parameters

Cloud service providers offer various types of VMs to the end user. We assume only five types of VMs with diverse specifications and computational capabilities similar to the implementation in the literature [41]. Additionally, the performance analytics offered by Amazon's Elastic Compute Cloud EC2is the basis of VMs [46]. The estimated processing time for each workflow task on multiple VM types is estimated based on their processing capacity. Furthermore, the performance variation of VMs in a single Data Center is modeled according to the direction of work presented in the literature [42] and it is similar to the literature [41,34, 43]. Nonetheless, the performance of VMs (with a variation of up to 24 percent) depends on the normal distribution, with a mean of 12 percent and a standard deviation of 10 percent. In addition, the data transfer time varies (up to 19 percent) according to a normal distribution with an average of 9.5 percent and a standard deviation of 5 percent. We assumed that our pricing model is similar to the current pricing model provided by Amazon [47], Google [48], and CloudSigma [44]. The average bandwidth for transferring intermediate I/O files across VMs is similar to the average bandwidth provided by the

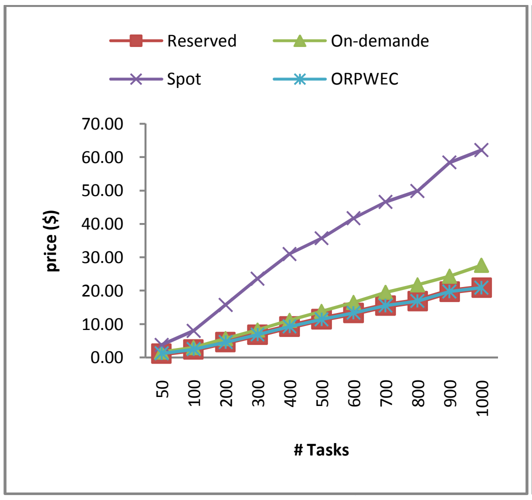
Amazon elastic block store (EBS) (20kbps) [45]. The billing time for our experience is one hour. We consider the structure of workflows and the execution time of tasks on the basis of DAX Directed Acyclic Graph in XML) [40] which are already known. Three sizes of scientific workflows were considered in our small, medium and large experiments. Each experiment was conducted ten times.

4.4 Results





Prices Cyberstake **Prices Montage**



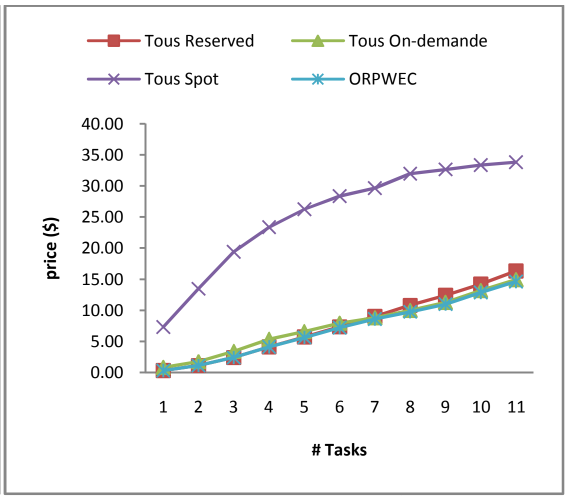


Fig. 6 Number of VMs vs number of tasks for different workflows

As illustrated in Figure 1, we simulate three workflows (CyberShake, Montage, Sipht). We can notice that the number of VMs used increases with the number of tasks. Each workflow has a different convergence relationship, such as; CyberShake approximatively has a linear relationship. However, Montage has an exponential relation. Since each workflow has a dependency on different tasks where some have more parallelization and others are sequential.

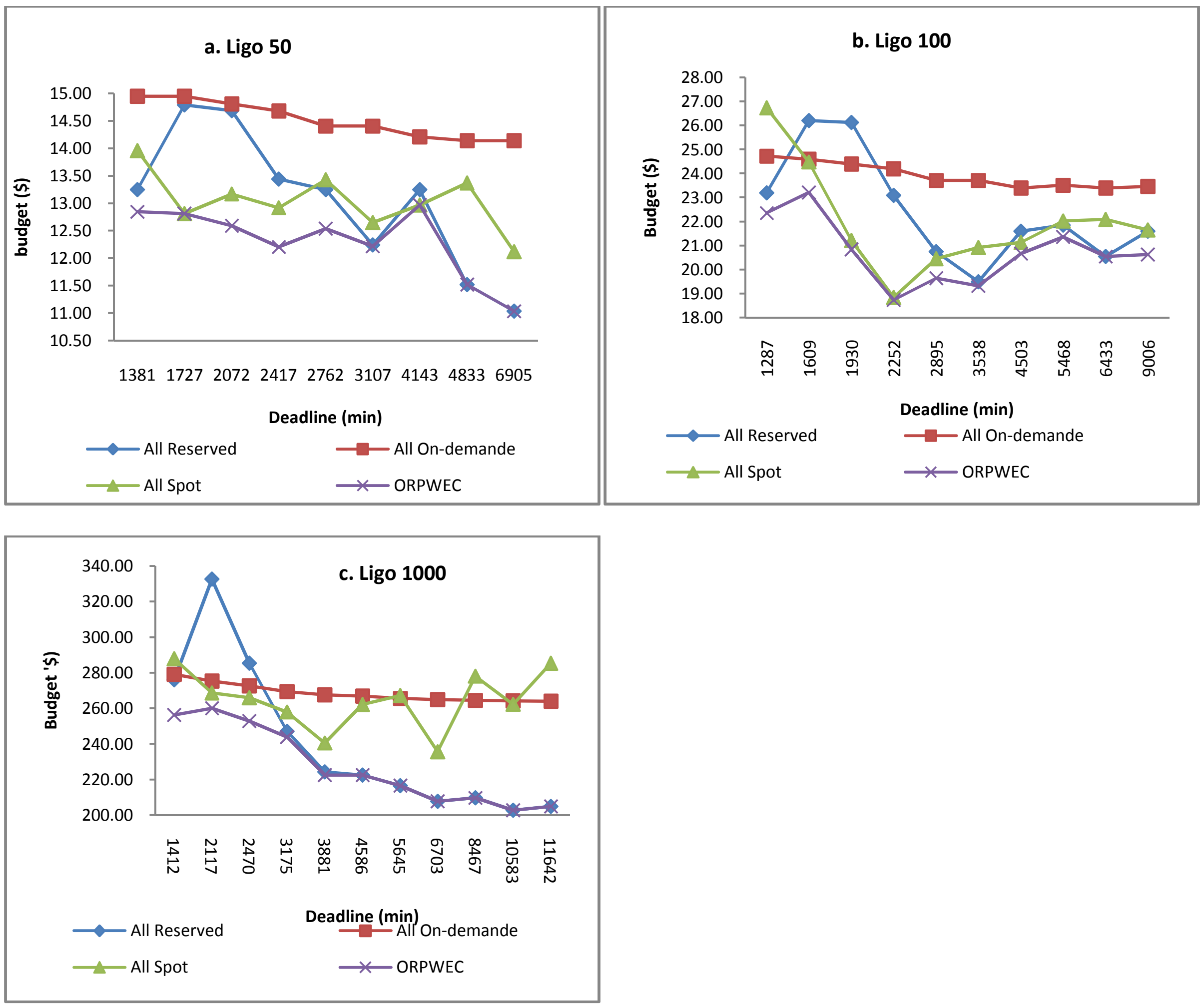


Fig. 7 Number of VMs and Budget vs deadline for LIGO workflow

In Figure 2, results differ from the previous results since we have varied the deadline for the same workflow (we have chosen Ligo) with different task sizes (50,100, and 1000). Figure 2.a demonstrates clearly that the number of used VM decrease as the deadline increase. This gives more ability to reduce the number of minimal VM required executing all workflow tasks respecting the defined deadline. Figure 2.b shows the proposed ORPWEC budget against all other budget types (on request, spoted, reserved). We can notice that ORPWEC reduces the budget by 5% from the lowest of the other types when the delay is low (more critical). However, as the deadline increase, the required budget will be approximatively equal to the reserved type due to the rate of the used VMs that will increase significantly. Therefore, this will make other budget types more expensive if they are based on an on-demand mechanism. Having shown some results of our proposed ORPWEC, we are now conducting a comparative study of ORPWEC to two other methods: Round Robin (RR) and First Come First Served (FCFS).

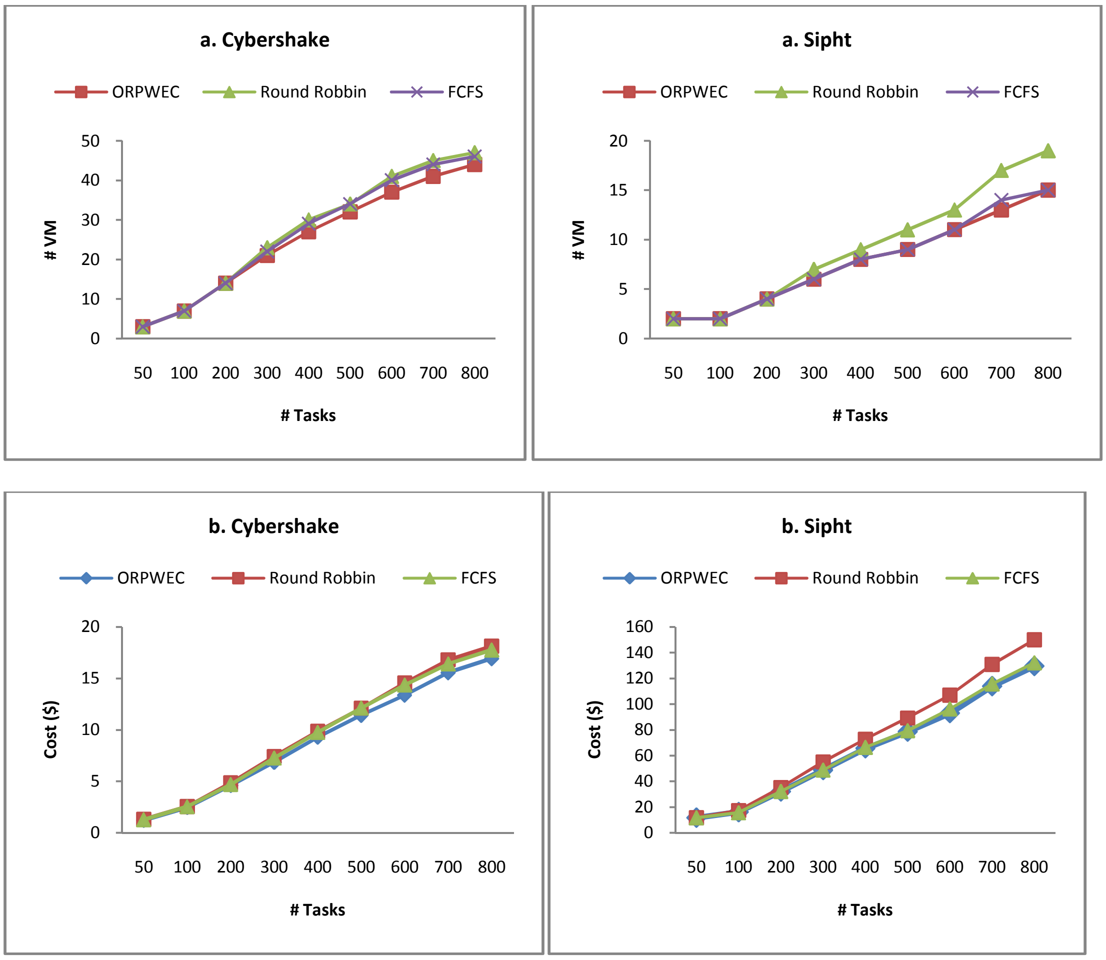


Fig. 8 Number of VMs and Budget vs number of tasks comparison

Figure 3 shows a performance comparison of both, the number of VMs and the budget constraint at different workflow sizes for each individual workflow. In Figure 3.a, we can observe that the number of VMs in ORPWEC is the same compared to RR and FCFS with CyberShake with a smaller size. However, ORPWEC reduces the number of VMs by up to 10% as the workflows grow. We can also notice that all algorithms give the same result for assembly due to the peculiarity of this type of workflow (more parallelism) that parallelized tasks take longer than others. Figure 3.b shows the budget consumption versus the number of workflow tasks for each algorithm. The same observation notices that the VM number in the montage where all algorithms have the same budget. Still, ORPWEC reduces the cost of CyberShake by about 5% at the smallest workflow size, increasing to 8% less loss than FCFS and 10% less loss than RR at the largest workflow size. For Sipht, ORPWEC and FCFS have the same budget, but with fewer workflow tasks, it is 10% less compared to RR. However, as the number of workflow tasks increases, ORPWEC provides better results reducing budget by 15% compared to RR and 3% compared to FCFS.

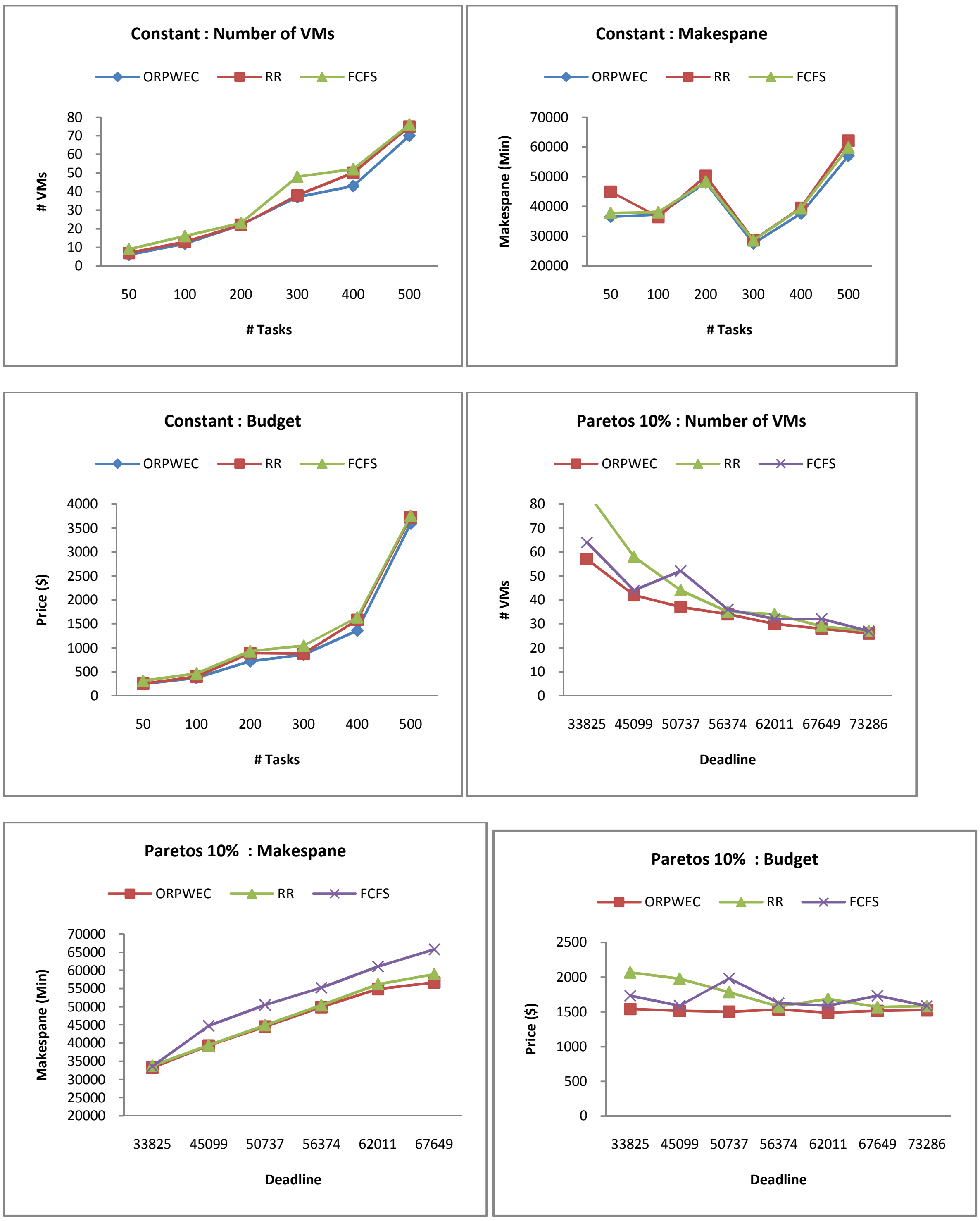


Fig. 9 Number of VMs and Budget vs number of tasks comparison for workflow ensemble

Figure 4 presents the results of workflow ensembles across different workflow sizes (constant, Pareto 10%). This result shows that ORPWEC outperforms both RR and FCFS. Figure 4.a shows the number of VMs in all scenarios. ORPWEC reduces the number of VMs by 4% for the smallest workflow ensemble and increases by up to 10% compared to RR and 20% compared to FCFS for Pareto scenarios. ORPWEC significantly reduces the budget by at least 10% compared to RR and 20% compared to FCFS in most scenarios. These results indicate that we used more workflow combinations and increased the number of workflow tasks. ORPWEC achieves better performance by minimizing the number of VMs and budgets, especially when deadlines are more critical (short deadlines).

5 Conclusion

In this paper, we address the interesting and critical new problem of scheduling and resource provisioning for scientific workflow ensembles in the cloud computing environment. In this paper, we deals with the challenge of resource provisioning for large-scale scientific workflows running on many heterogeneous clouds by running all processes of the ensemble under budget and time constraints. ORPWEC is the proposed model based on four algorithms. The algorithms were evaluated via simulation on ensembles of real workflows (Cyberstake, Montage, Sipht, Ligo). ORPWEC used a minimal number of VMs and has decreased the budget by 5%. After that, we have compared ORPWEC with RR and FCFS. ORPWEC has reduced the number of VMs by up 10%. ORPWEC reduces the cost of CyberShake by about 5% at the smallest workflow size, increasing to 8% less loss than FCFS and 10% less loss than RR at the largest workflow size.

References

[1] T. Coleman, H. Casanova, L. Pottier, M. Kaushik, E. Deelman, R. F. da Silva, “WfCommons: A framework for enabling scientific workflow research and development”, Future Generation Computer Systems, Vol. 128, 2022, pp. 16-27,

[2] G. Juve, A. Chervenak, E. Deelman, S. Bharathi, G. Mehta, and K. Vahi, ``Characterizing and profiling scientific workflows", *Future Generation Computer Systems*, Vol. 29, No. 3, 2013, pp. 682-692.

[3] J. C. Jacob and D. S. e. a. Katz, “Montage: a grid portal and software toolkit for science-grade astronomical image mosaicking”, *International Journal of Computational Science and Engineering*, Vol. 4, No. 2, 2009, pp. 73–87.

[4] A. Abramovici, W. E. Althouse, and et. al., “LIGO: The laser interferometer gravitational-wave observatory,” *Reports on Progress in Physics*, Vol. 256, No. 5055, 1992, pp. 325–333.

[5] R. Graves, T. H. Jordan, and et. al., “Cybershake: A physics-based seismic hazard model for Southern California”, *Pure and Applied Geophysics*, Vol. 168, No. 3-4, 2010, pp. 367–381.

[6] Z. Ahmad et al., "Scientific Workflows Management and Scheduling in Cloud Computing: Taxonomy, Prospects, and Challenges" *IEEE Access*, Vol. 9, 2021, pp. 53491-53508.

[7] V. Arabnejad, K. Bubendorfer and B. Ng, "A Budget-Aware Algorithm for Scheduling Scientific Workflows in Cloud", *2016 IEEE 18th International Conference on High Performance Computing and Communications;* Sydney, NSW, Australia, 2016, pp. 1188-1195.

[8] O. H. Ibarra and C. E. Kim, ``Heuristic algorithms for scheduling independent tasks on non identical processors", *Journal of the ACM*, Vol. 24, No. 2, 1977, pp. 280-289.

[9] M. R. Garey and D. S. Johnson, *Computers and Intractability; A Guide to the Theory of NP-Completeness*. New York, NY, USA: Freeman, 1979.

[10] M.A. Rodriguez, *“Budget-Driven Resource Provisioning and Scheduling of Scientific Workflow in IaaS Clouds with Fine-Grained Billing Periods.”* (2016).

[11] Y. Gao, S. Zhang and J. Zhou, "A Hybrid Algorithm for Multi-Objective Scientific Workflow Scheduling in IaaS Cloud", *IEEE Access*, Vol. 7, 2019, pp. 125783-125795.

[12] M. Mao and M. Humphrey, “Auto-scaling to Minimize Cost and MeetApplication Deadlines in Cloud Workflows”, *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, Seattle, WA, USA, 2011, pp. 1-12.

[13] R. Calheiros and R. Buyya, “Meeting Deadlines of Scientific Workflows in Public Clouds with Tasks Replication”, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 25, No. 7, 2014, pp. 1787-1796.

[14] A. C. Zhou, B. He, and C. Liu, “Probabilistic Scheduling of Scientific Workflows in Dynamic Cloud Environments,” *CoRR*, 2013, Vol.abs/1306.6410.

[15] T. A. L. Genez, L. F. Bittencourt, and E. R. M. Madeira, “Workflow Scheduling for SaaS/PaaS Cloud Providers Considering Two SLA Levels”, *Network Operations and Management Symposium*, Maui, HI, USA, 2012, pp. 906 –912.

[16] S. Pandey, L. Wu, S. Guru, and R. Buyya, “A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments”, *24th IEEE Int'l Conf. on Advanced Information Networking and Applications*, April 2010, pp. 400–407.

[17] H. Topcuoglu, S. Hariri, and M.-Y. Wu, “Performance-effective and low-complexity task scheduling for heterogeneous computing” *IEEE Transactions on Parallel and Distributed Systems*, 2002, Vol. 13, No. 3, pp. 260–274.

[18] A. Asghari, M.K. Sohrabi, and F. Yaghmaee, F., “Task scheduling, resource provisioning, and load balancing on scientific workflows using parallel SARSA reinforcement learning agents and genetic algorithm”, *Journal of Super computing,* 2021, Vol. **77**, pp. 2800–2828.

[19] E.-K. Byun, Y.-S. Kee, J.-S. Kim and S. Maeng, “Cost optimized provisioning of elastic resources for application workflows”, *Future Generation Computer Systems*, 2011. Vol. 27, No. 8, pp. 1011–1026.

[20] T. T. Huu and J. Montagnat, “Virtual Resources Allocation for Workflow-Based Applications Distribution on a Cloud Infrastructure,” *10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, 2010, pp. 612–617.

[21] J. Chen, C. Wang, B. B. Zhou, L. Sun, Y. C. Lee, and A. Y.Zomaya, “Tradeoffs between profit and customer satisfaction for service provisioning in the cloud”, *Proceedings of the20th International Symposium on High Performance Distributed Computing (HPDC'11)*, ACM, San Jose, Calif, USA,2011, pp. 229–238.

[22] H. Kim, Y. El-Khamra, I. Rodero, S. Jha, and M. Parashar, “Autonomic management of application workflows on hybrid computing infrastructure,” *Scientific Programming*, 2011, Vol. 19, No.2-3, pp. 75–89.

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

[24] R. Duan, R. Prodan, X. Li, “A sequential cooperative game theoretic approach to Storage-Aware scheduling of multiple Large-Scale workflow applications in grids”, *Grid Computing (GRID), 2012 ACM/IEEE 13th International Conference on, IEEE,* 2012, pp. 31-39.

[25] R. Duan, R. Prodan, X. Li, “Multi-objective game theoretic scheduling of bag-of-tasks workflows on hybrid clouds”, *IEEE Transactions on Cloud Computing*, 2014, Vol. 2, No 1, pp. 29-42.

[26] P.A. Malla, S. Sheikh, “Analysis of QoS aware energy-efficient resource provisioning techniques in cloud computing”, 2023, *International Journal of Communication Systems*, Vol. *36, No* 1. e5359.

[27] M. Malawski, G. Juve, E. Deelman, and J. Nabrzyski, “Cost-and deadline-constrained provisioning for scientific workflow ensembles in IaaS clouds,” *Proceedings of Int'l Conference on High Performance Computing, Networking, Storage and Analysis*, 2012, pp. 22:1–22:11.

[28] P. Bryk, M. Malawski, G. Juve, and E. Deelman, “Storage-aware Algorithms for Scheduling of Workflow Ensembles in Clouds”, *Journal of Grid Computing*, 2016, Vol. 14, No. 2, pp. 359-378.

[29] A.C. Zhou, B. He, X. Cheng, and C.T. Lau, “A Declarative Optimization Engine for Resource Provisioning of Scientific Workflows in IaaS Clouds”, *Proceedings of the 24th International Symposium on High-Performance Parallel and Distributed Computing* (HPDC '15). ACM, New York, NY, USA, 2015, pp. 223-234.

[30] T. A. L. Genez, L. F. Bittencourt, R. Sakellariou and E. R. M. Madeira, "A Flexible Scheduler for Workflow Ensembles", *IEEE/ACM 9th International Conference on Utility and Cloud Computing (UCC), Shanghai, China*, 2016, pp. 55-62.

[31] J. Varia, “Cloud Computing: Principles and Paradigms, chapter Architecting Applications for the Amazon Cloud”, Wiley Press, 2011, pp. 459–490.

[32] Amazon Inc. Amazon Elastic Compute Cloud (Amazon EC2). http://aws.amazon.com/ec2.

[33] O. Agmon Ben-Yehuda, M. Ben-Yehuda, A. Schuster and D. Tsafrir, "Deconstructing Amazon EC2 Spot Instance Pricing", 2013, *ACM Transactions on Economics and Computation*, Vol. 1, No 3, pp. 1–20.

[34] M. A. Rodriguez and R. Buyya, "Deadline Based Resource Provisioningand Scheduling Algorithm for Scientific Workflows on Clouds,", 2014, *IEEE Transactions on Cloud Computing*, Vol. 2, No. 2, pp. 222-235.

[35] W. Chen and E. Deelman, "WorkflowSim: A toolkit for simulating scientific workflows in distributed environments", 2012, *IEEE 8th International Conference on E-Science, Chicago, IL, USA*, pp. 1-8.

[36] Montage: An Astronomical Image Mosaic Engine, accessed on June. 28, 2023. [Online]. Available: http://montage.ipac.caltech.edu .

[37] D. A. Brown, P. R. Brady, A. Dietz, J. Cao, B. Johnson, and J. McNabb, ‘‘A case study on the use of workflow technologies for scientific analysis: Gravitational wave data analysis’’, 2007, *Workflows for e-Science. London, U.K.: Springer*, pp. 39–59.

[38] E. Deelman, D. Gannon, M. Shields, and I. Taylor, ‘‘Workflows and e-science: An overview of workflow system features and capabilities’’, 2009, *Future Generation Computer Systems*, Vol. 25, No. 5, pp. 528–540.

[39] S.S. Murad, R. Badeel, N. Salih, A. Alsandi, R. Faraj, A.R. Ahmed, A. Muhammed, M. Derahman, N. Alsandi, “Optimized Min-Min task scheduling algorithm for scientific workflows in a cloud environment”, 2022, *Journal of Theoretical and Applied Information Technology*, Vol. 100, No 2 pp. 480–506.

[40] T. Coleman, H. Casanova and R. da Silva, "WfChef: Automated Generation of Accurate Scientific Workflow Generators", 2021, *IEEE 17th International Conference on eScience (eScience), Innsbruck, Austria*, pp. 159-168.

[41] J. Sahni and D. P. Vidyarthi, "A Cost-Effective Deadline-Constrained Dynamic Scheduling Algorithm for Scientific Workflows in a Cloud Environment", 2018, *IEEE Transactions on Cloud Computing*, Vol. 6, No. 1, pp. 2-18.

[42] J. Schad, J. Dittrich, and J.-A. Quiané-Ruiz, ‘‘Runtime measurements in the cloud: Observing, analyzing, and reducing variance’’, 2010, *Proc. VLDB Endowment*, Vol. 3, No 1–2, pp. 460–471.

[43] N. A. binti Muhamad Shaari, T. F. Ang, L. Y. Por, and C. S. Liew, “Dynamic Pricing Scheme for Resource Allocation in Multi-Cloud Environment”,2017, *MJCS*, Vol. 30, Vo. 1, pp. 1–17.

[44] CloudSigma, accessed on June. 28, 2023. [Online]. Available: https://www.cloudsigma.com/pricing.

[45] Amazon Elastic Block Store (Amazon EBS), accessed on June. 28, 2023. [On-line]. Available: http://aws.amazon.com/ebs/.

[46] S. Ostermann, A. Iosup, N. Yigitbasi, R. Prodan, T. Fahringer, and D. Epema, ‘‘A performance analysis of EC2 cloud computing services for scientific computing’’, 2010, *First International Conference Cloud Computing*. Berlin, Germany: Springer, pp. 115–131.

[47] *Amazon EC2 Pricing*, accessed on June. 28, 2023. [Online]. Available: https://aws.amazon.com/ec2/pricing/.

[48] *Google Cloud Platform*, accessed on June. 28, 2023. [Online]. Available: https://cloud.google.com/compute