View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Institute of Advanced Engineering and Science

International Journal of Electrical and Computer Engineering (IJECE)
Vol. 8, No. 2, April 2018, pp. 853~866
ISSN: 2088-8708, DOI: 10.11591/ijece.v8i2.pp853-866

Workflow Scheduling Techniques and Algorithms in IaaS Cloud: A Survey

K. Kalyana Chakravarthi, Vaidehi Vijayakumar
School of Computing Sciences and Engineering, VIT University, Chennai, India

Article Info

Article history:

Received Oct 26, 2017
Revised Dec 27, 2017
Accepted Jan 11, 2018

Keyword:

IaaS cloud
Meta-heuristics
Resource provisioning
Scientific workflows
Workflow scheduling

ABSTRACT

In the modern era, workflows are adopted as a powerful and attractive paradigm for expressing/solving a variety of applications like scientific, data intensive computing, and big data applications such as MapReduce and Hadoop. These complex applications are described using high-level representations in workflow methods. With the emerging model of cloud computing technology, scheduling in the cloud becomes the important research topic. Consequently, workflow scheduling problem has been studied extensively over the past few years, from homogeneous clusters, grids to the most recent paradigm, cloud computing. The challenges that need to be addressed lies in task-resource mapping, QoS requirements, resource provisioning, performance fluctuation, failure handling, resource scheduling, and data storage. This work focuses on the complete study of the resource provisioning and scheduling algorithms in cloud environment focusing on Infrastructure as a service (IaaS). We provided a comprehensive understanding of existing scheduling techniques and provided an insight into research challenges that will be a possible future direction to the researchers.

Copyright © 2018 Institute of Advanced Engineering and Science.
All rights reserved.

Corresponding Author:

K. Kalyana Chakravarthi,
School of Computing Sciences and Engineering,
VIT University,
Chennai, Tamil Nadu - 600 127, India.
Email: kalyan.koneti@gmail.com

1. INTRODUCTION

Scientific workflow management systems are becoming popular for solving the problems which involve complex data of different size, data analysis, and higher processing power. A process can be modeled as a workflow by dividing it into smaller sub-processes, called tasks. These sub-processes are distributed to multiple computing resources for faster parallel execution. The resource provisioning and scheduling problems have been widely studied in the literature from homogeneous clusters to the most recent paradigm, cloud computing. This work focuses on the complete study of the resource provisioning and scheduling algorithms in cloud environment focusing on Infrastructure as a service (IaaS).

Different cloud providers have various product offerings such as Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS). However, this survey focuses on IaaS clouds. The IaaS clouds provide access to a virtual pool of unlimited and heterogeneous resources on-demand, which are flexible and scalable for deploying resource-intensive scientific workflows. This enables the workflow management systems to access the resources on-Demand with a predefined configuration of a virtual machine (VM). These VMs can be acquired or released on-Demand and being possessed in fine grained pricing unit, for example, Amazon charges per hour and Azure charged on per minute basis.
Previous research works for clusters and grids focused on:

a. Meeting deadlines or minimizing the makespan without considering the infrastructure cost.

b. Focus only on task-resource mapping stage as the resources where configurations are known in advance. Where as in Cloud another stage exists called resource provisioning to determine the number and type of resources required which affects both makespan of the workflow execution and the cost.

In this survey, various characteristics of existing resource provisioning and scheduling algorithms are analyzed. To achieve performance and cost objectives, Schedulers should be aware of the dynamic nature of the cloud platforms and the uncertainties in resources such as the performance of VM, network bandwidth, provisioning and de-provisioning VMs. We extended our discussion focused on the cloud which includes dynamic resource provisioning and scheduling decisions, Quality of Service (QoS), Quality of Data (QoD) constraints, scheduling objectives and Optimization strategies.

1.1. Cloud Resource Model

The resource model defines the components of processing elements and connections among processing elements in parallel computing environment [1]. In IaaS, cloud resource is called a (instantiated as) VM. A VM is an environment that is installed on software that behaves as if it is a separate OS. Different cloud providers provide different VM types of VT = {vt1, vt2, ……., vt|VT|}, where vt is associated with two attributes of capability and cost.

1.2. Resource access and pricing models

Interface and Infrastructure together forms a cloud environment [2]. Various IaaS providers offer diverse resource access and pricing options for reserved, on-demand and spot instances.

a. Reservedinstance. This model allows the users to reserve computing resources for a longer period by prepaying the reservation fee.
b. On-demand instance. This model allows the users to lease the resource in a fine-grained manner based on the demand. On-demand instances are usually charged more than the reserved instances for the same VM type. On-demand option adopts coarse-grained billing option, for example, Amazon EC2 charges per hour billing cycle, whereas Azure charges per minute billing cycle. Partial usage of instances is rounded up to next billing cycle.
c. Spot instance. Some of the IaaS providers offer dynamic pricing scheme for their large quantities of spare capacity unsold. These dynamically priced VMs are called as spot instances, whose price changes based on the market supply and demand patterns. This model allows the users to bid on those spare capacities and grants the resources to the users if their bid price is higher than the spot instance price. Spot instance prices are usually much lower than the on-demand instance price. However, spot instance may be terminated at any time when the bidding price is lower than the spot instance price.

Rest of the paper is as follows. Section 2 introduces the overview of scientific workflows and application model taxonomy. Section 3 discusses workflow scheduling techniques based on the literature survey for both resource and workflow. A detailed discussion of outstanding algorithms in Section 4, Finally, research challenges in Section 5 and conclusion are described in Section 6.

2. OVERVIEW OF SCIENTIFIC WORKFLOWS

Workflow is the series of orchestrated activities that are necessary to complete a task. It has its roots in commercial enterprises of office automation systems as a business processing tool, which is a matured research area [3]. Scientific workflows describe a series of computations that enable the analysis of data in a structured and distributed manner and allows scientists to model, design, execute, debug, re-configure and re-run analysis and visualization processes.

Workflows can be expressed and processed as best effort, superscalar, and streaming pipelines. We focus on Directed acyclic graphs (DAGs) as they are commonly used by scientific and research community. DAGs by definition do not have cycles or control dependencies. For example, workflow management systems such as Pegasus, CloudbusWfMS, ASKALON, and DAGMansupport the workflows modeled as DAGs.

Formally, a DAG can be represented as W = G(T, E),where

T is a set of v =| T| tasks $\{t_1,t_2,...,t_n\}$ and each task is a sequence of instructions that must be executed,

And

Eis a set of e =| E| directed edges $\{e_{ij} |(t_i,t_j) \in E\}$

Representing inter-task data dependencies, in which ti is said to be the parent task of tj and tj is said to be child task of ti. Each edge e_{ij} represents a precedence relation which indicates that task tj should not start its executionuntil task ti execution is completed and its input data is available to tj. A task which does not have parent task is called an entry task, where as a task which does not have child task is called an exit task.To generalize the DAG structure with only one entry task and one exit task, two dummy tasks t_{start} and t_{end} are usually adopted and added to the begin and end of the workflow, which have zero computation workload and zero communication data to actual start tasks and end tasks.

The process of scheduling a workflow represents assigning tasks to resources and orchestrating their execution to preserve the dependencies among the tasks. Formally, a schedule [4], S is represented as a 3-tuple, S= <R, M, makespan>, Where R is the set of resources $\{r_1, r_2, r_3 \ldots r_n\}$ and M consists of task resource mappings. The makespan is the schedule length of S.

All the algorithms in this survey differ in their ability to schedule in the workflow multiplicity. Algoriths are designed to schedule different types of workflows with different multiplicity like a Single nstance of workflows, or multiple instances of the same workflows, multiple workflows. Rodriguez and Raj [5] identified 3 types of scheduling processes from the workflow multiplicity perspective.

a. Single Workflow
   Single workflows are executed sequentially and independently. This is the traditional model used in grids, clusters, and clouds. Algorithms in this class are focused on cost optimization and meeting the QoS [6] requirements for a single user and a single DAG.
b. Workflow Ensembles
   Workflow ensembles are interrelated workflows which are grouped together. These workflows will have the similar structure but differ in their size and input data. Algorithms in this class are focused on executing all the instances of the workflow in the ensemble with available resources. QoS requirements are only specified for multiple workflows but not for single workflow. In this model, a number of instances are known in advance and scheduler use this for planning and execution of tasks.
c. Multiple Workflows
   Multiples workflows are similar to ensembles except that, the workflows scheduled may not be related to each other and also vary in size, structure, and data. Scheduling is viewed as dynamic as the number and type of workflows are unknown in advance. Each independent workflow will have its own QoS requirement.

3. TAXONOMY OF WORKfIOW SCHEDULING

Workflow scheduling problem has been studied extensively over past years and many heuristics have been developed for scheduling the tasks in distributed environments. The input to the workflow scheduling algorithm is the *abstract workflows* which define the tasks without specifying the location of the resources in which these tasks are executed. Abstract workflows are categorized as *deterministic* and *non-deterministic.* In the deterministic model, task dependencies and input data are known in advance, whereas in non-deterministic model they are known at the run time only.

Workflow scheduling can be categorized as best-eff ort based and QoS constraint-based scheduling [7] as shown in Figure 1. The Best-effort scheduling focuses on minimizing the makespan, ignoring various user's QoS constraints. The QoS constraint-based scheduling attempts to minimize the performance under most important QoS constraints, for example, minimize cost under deadline constraints or minimize time under budget constraint.

3.1 Best-Effort based Scheduling

The best-effort scheduling attempts to optimize one objective while ignoring other factors such as monetary cost and various QoS requirements. The objective of best-effort scheduling algorithms is to minimize the makespan. The makespan of a workflow application is the total time taken to complete the execution of a workflow.

Best-Effort scheduling algorithms are either *heuristics* based or*meta-heuristics* based approach. The heuristic based algorithms fit only to the particular type of problems while meta-heuristics based algorithms are based on a meta-heuristic method which provides the general solution for developing a specific heuristic to fit into a specific problem.

3.1.1. Heuristics

There are four classes of scheduling heuristics for a workflow application. Individual task scheduling, list scheduling, cluster and duplication based scheduling.

3.1.1.1. Individual/Immediate Task Scheduling

Individual task scheduling is a simple scheduling method, which makes the decision on an individual task. TheMyopic algorithm implemented in Condor DAGMan schedules the unmapped ready task to a resource which can complete the task earliest. This step is repeated until all tasks are scheduled.

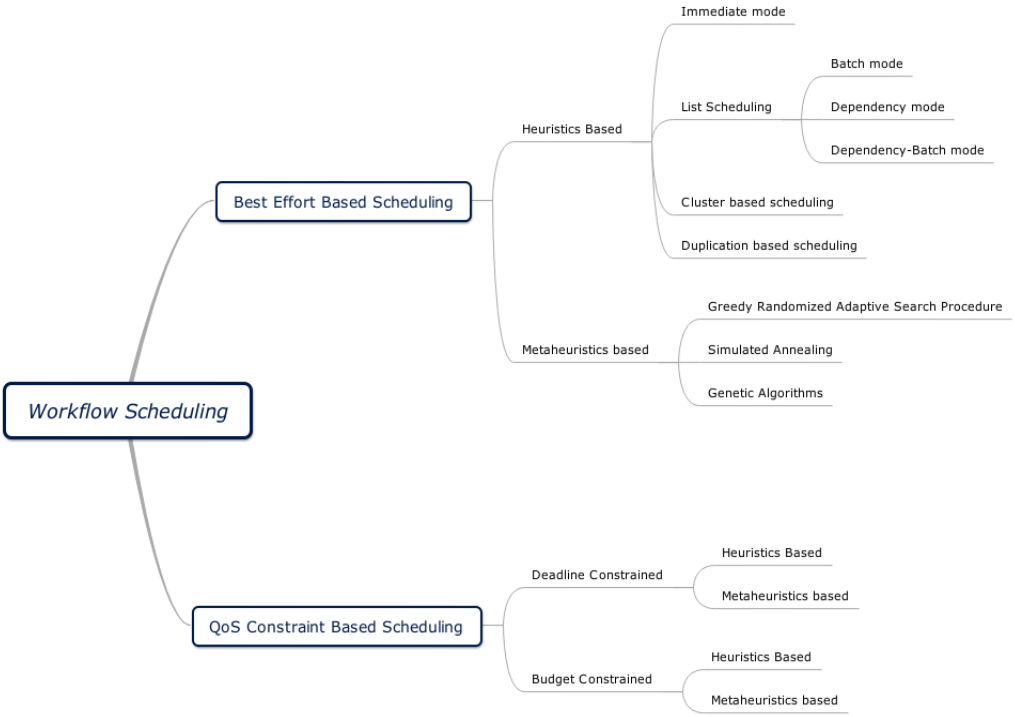

Figure 1. Taxonomy of workflow scheduling Algorithms

3.1.1.2. List Scheduling

List scheduling generates a scheduling list of tasks based on their priorities with a rank value and sorts the list according to their rank values, and execute the following two steps repeatedly until all tasks in the DAG are scheduled.

a. Task prioritizing Sets the priority to each task with a rank value from the scheduling list;
b. Resource Selection Selects the task in the order of priority and allocates the task to the optimal resource.

The scheduling list can be constructed either statically or dynamically. If all the priorities of tasks are constructed before any task allocation, is called as static list scheduling. Whereas if the priorities of unscheduled tasks are recomputed after each task scheduling step, it is called as dynamic task scheduling. Whether static or dynamic list, different prioritizing attributes and resource selection strategies are required to decide task priorities and optimal resource for each task. Workflow list scheduling algorithms are either batch, dependency or dependency-batch mode.

3.1.1.2.1. Batch Mode

Batch mode scheduling algorithms group the tasks into several independent tasks, such as a bag of tasks and parameter tasks. It considers the current group tasks to complete the execution at the earliest time. Some classic batch mode heuristics are Min-Min, Max-Min, Suff erage proposed by Maheswaranetal [8].

3.1.1.2.2. Dependency Mode

Dependency mode scheduling algorithms are based on scheduling a task with interdependent task algorithms. Dependency mode intends to provide a strategy to map workflow tasks on heterogeneous resources based on analyzing the dependencies of the entire task graph. Unlike batch mode algorithms, it ranks all priorities of all tasks based on whole application context.

Topcuoglu et al. [9] proposed the Heterogeneous-Earliest-Finish-Time (HEFT) algorithm that calculates the average execution time for each task and average communication time between resources of two successive tasks. Then for each task, a rank value is calculated in a recursive manner based on the rank value of its dependent tasks. Exit task will have the lowest rank value as being the average execution time. The predecessors of the exit task will have their average execution time + the maximum ([communication time from a resource to another resource] + [the rank value of the successor]). Then task with the highest priority will be scheduled first.

3.1.1.2.3. Dependency-Batch Mode

Zhao and Sakellariou [10] proposed a hybrid heuristic approach for scheduling DAG in heterogeneous systems. This heuristic approach combines dependency mode and batch mode. It first computes the rank of each task and creates a group of independent tasks. Then it organizes tasks group by group and uses a batch mode algorithm to reprioritize the scheduling of tasks in the group.

Table 1 summarizes the typical list scheduling heuristics for heterogeneous earliest-finish-time (HEFT), critical-path-on-a-processor (CPOP), dynamic level scheduling (DLS), dynamic critical path (DCP).

Table 1. Typical list scheduling algorithms

Heuristic	Attribute	Static/Dynamic	Complexity
HEFT	$rank_u(ti)$	Static	O(e X |R|)
CPOP	$rank(ti) = rank_u(ti) + rank_d(ti)$	Static	O(e X |R|)
DLS	$DL(t_i, r_l) = rank_u(ti) - EST_{ti}^{rl}$	Dynamic	$O(V^3$ X |R|)
DCP	$LST(t_i) – EST(t_i)$	Dynamic	$O(V^3)$

3.1.1.3. Clustering Heuristic

Both clustering based and duplication based scheduling (in 3.1.1.4) are designed to optimize the data transmission time between dependent tasks. In most scheduling heuristics, tasks are scheduled separately. A schedule that does not consider the communication delay generates the idle fragments in the resource.

Clustering heuristic consists of two parts:

a. Clustering Mapping tasks to clusters;
b. OrderingOrdering tasks in the same cluster.

Clustering of a DAG is the mapping of all tasks on to clusters, where each cluster is a subset of tasks and executes on a separate resource, which minimizes the data transmission time between tasks. Hence there is a trade-off between maximizing the parallelism and minimizing the communication delay. If there are two independent tasks in the same cluster, it is called nonlinear cluster; otherwise called linear. A linear clustering preserves parallelism, hence it does not increase the parallel execution time. Whereas in nonlinear clustering, parallel tasks are sequentialized which reduces the parallelism and increases the parallel execution time.

Table 2 overview of clustering heuristics.

Table 2. Compares four typical clustering heuristics

Heuristic	Clustering	ordering	Linear	Complexity
DSC [11,12]	Chooses the free task with highest rank. Zero its incoming edges if it reduces $rank_d(ti)$	Non-insertion	No	O (log v (v + e))
KB/L [13]	Clustering all tasks on the current critical path	NA	Yes	O (v x (v + e))
Sarkar's [14]	Zero the highest communication edge if the makespan does not increase	Highest rank first	No	O (e x (v + e))
CASS–II [15]	Chooses the current task with highest rank	Non-insertion	No	O (e + (v x logv))

3.1.1.4. Duplication Heuristic

The duplication based heuristicis to duplicate task on the same resource with target task so that transmission time between those two tasks avoided. The motivation of duplication heuristic is, it may happen that some resources may be idle during different time periods because tasks assigned to them may be waiting for the data from other resources. These idle time slots are effectively utilized by identifying the critical tasks and allocating them in these slots, further reducing the execution time. Based on the selection of tasks to be duplicated, duplication-based algorithms are divided into two classes; scheduling with full duplication (SFD) and scheduling with partial duplication (SPD). The SFD duplicated all tasks from higher levels of target tasks, whereas SPD imposes restrictions during task selection process for duplication. Duplication-based scheduling achieves shorter makespans, it also makes the scheduling more difficult. Table 3 gives an overviewof duplication heuristics.

Table 3. Overview of duplication heuristics

Heuristic	Feature	List\Clustering	Duplication type
TDS [16]	Critical parent is duplicated	Clustering	Partial Duplication
LWB [17]	Lower bound of start time is approximated	Clustering	Partial Duplication
PLW [18]	Lower bound of start time is approximated	Clustering	Partial Duplication
DFRN [19]	Duplication first and reduction next	List	Partial Duplication
DSH [20]	Utilization of idle slot is maximized	List	Full Duplication
BTDH [21]	Extension of the DSH	List	Full Duplication
LCTD [22]	Optimization of linear clustering	Clustering	Full Duplication
CPFD [23]	Task on critical path is considered first	List	Full Duplication
PY [24]	Lower bound of start time is approximated	Clustering	Full Duplication
TCSD [25]	Lower bound of start time is approximated	Clustering	Full Duplication

3.1.2. Meta-heuristics

As DAG scheduling is an NP-completeproblem, an optimal solution can't be found in polynomial time unless NP = P [26]. Meta-heuristics usually provide the general structure and strategy guidelinesfor developing heuristic. In literature, few meta-heuristics are proposed, which provides an efficient way of moving quickly toward a very good solution.

Genetic Algorithms (GAs) are most widely studied meta-heuristic, which provides the optimal solution for large search space using the principle of evolution. These GAs differ in the representation of the schedules in the search space, genetic operators for generating new schedules, fitness function to evaluate the schedules and stochastic assignment to control the genetic operators. Literature survey shows that GA-based approach requires around one minute to produce a solution, while other heuristics require an execution of few seconds.

Greedy Randomized Adaptive Search Procedure (GRASP) is a randomized iterative search technique. Blythe et al. [27] investigated GRASP, which gets better performance than Min-Min heuristic on both computational- and data-intensive applications. Simulated Annealing (SA) [28] is motivated by the Monte Carlo method for statistically searching the global optimal between different local optima. Young et al. [29] have investigated performances of SA algorithms for scheduling workflow applications in a Grid environment.

3.1.3. Comparison of Best-Effort Scheduling Algorithms

Meta-heuristics perform better than local search based heuristics as they search schedules in larger search space. However, when the number of tasks in the workflow DAG increases, scheduling time of meta-heuristics increases rapidly. List scheduling heuristics assumes a bounded number of resources whereas clustering heuristics assumes the unbounded number of resources.

3.1.4. Dynamic Workflow Scheduling

The performance of resources varies over time. A 'best' resource may become the 'worst' resource or vice-versa. Sonmez et al. [30] proposed a taxonomy of dynamic scheduling policies based on resource information (status, processing speed, and link speed) and task information (task length and communication

data size). Dynamic scheduling is developed for handling the unavailability of the resource and task information. Dynamic algorithms make the task to VM assignment decisions during runtime. These decisions are based on current state of the resource and task information.

3.2. QoS-Constraint Based Workflow Scheduling

Most of the QoSconstraint-based workflow scheduling heuristics are based on *time* or *cost*. Time is the total time taken to execute all tasks of the workflow (deadline). Cost is the total expense for executing the workflow (budget). Scheduling algorithms based on time and cost constraints, called Deadline constrained scheduling and budget constrained scheduling.

3.2.1. Deadline Constrained Scheduling

Deadline constrained scheduling aims to minimize the execution cost while meeting the user specified deadline constraint. In literature survey, two heuristics have been developed to minimize the cost while meeting the time constraint. *Back-tracking*proposed by Menasc and Casalicchio [31] and *Deadline Distribution*proposed by Yu et al. [32].

In*backtracking* heuristic, available or unmapped tasks are assigned to the least expensive resource. In case if many tasks are available, it assigns most data intensive task to a fastest resource. This heuristic is repeated until all tasks are assigned to a resource. After each iterative step, the execution time of current assignment is computed, if it exceeds the time constraint, the heuristic backtracks the previous step, removes the least expensive resource from its list and reassigns the tasks with updated resource list.

In deadline distribution heuristic, the workflow is partitioned and distributes the overall deadline to each task based on their workload and dependencies. Workflow tasks are clustered into partitions and deadline is distributed over each partition, then each partition is assigned with a deadline. For each task in the partition, a sub-deadline can also be assigned.

3.2.2. Budget Constrained Scheduling

Budget constrained scheduling aims to minimize workflow execution time while meeting users' specified budgets. Tsiakkouri et al. [33] proposed budget constrained scheduling called LOSS and GAIN. This scheduling adjusts a schedule generated by a time optimized heuristic and a cost optimized heuristic to meet users' budget constraints. A time optimized heuristic attempts to minimize execution time while a cost optimization attempts to minimize execution cost.

If Total execution cost generated by time optimized schedule is greater than the budget; the LOSS approachis applied. If the Total execution cost generated by a cost optimized schedule is less than the budget, the GAIN approach is applied in order to use the surplus for decreasing theexecutiontime.

4. SURVEY

This section describes a set of existing algorithms and depicts a complete classification with a focus on IaaS clouds. Results are summarized in Table 4.

4.1. Multilevel Deadline-constrained Scientific Workflows

Malawski et al. [34] proposed a cost-optimization model for scheduling scientific workflows in IaaS clouds, using mathematical programming languages (AMPL and CMPL) which optimizes the cost under a deadline constraint in multi-cloud environment, where each provider offers a limited number of heterogeneous VMs, and cloudobjectstoresuchasAmazonS3 to share intermediate data files. Their method proposes different models such as application model, infrastructure model, and the scheduling model as mixed integer programming (MIP). Two different scheduling models are proposed, one for Coarse-grained tasks in which tasks have average run time in the order of one hour, and other fine-grained tasks with deadlines shorter than one hour. This model takes the advantage of some of the characteristics of scientific workflows such as sequential levels of independent tasks. In each level, a set of tasks are partitioned into several groups based on their computational cost and input/output data size with a constraint of instances that cannot be shared in multiple levels, may lead to low resource utilization and high resource cost. Potential improvement could be to study the possibility of scheduling for each level that can be computed in parallel.

4.2. Security-Aware and Budget-Aware (SABA)

SABA algorithm [35] focuses on minimizing the makespan of a scientific workflow with user's security constraint under budget constraints as well as security in multi-cloud environment. They proposed two types of data sets, immovable and movable datasets. Movable data sets do not have any security constraints, hence can be migrated or replicated from one data center to another data center. Immovable

datasets are restricted to one datacenter, and cannot be migrated or replicated from one datacenter to another datacenter because of security and cost constraints. The SABA algorithm consists of 3 phases. In the first phase, clustering and prioritization phase in which data and tasks are assigned to a data center using priority rank. In the second phase, tasks are assigned to VMs on the basis of performance-cost ratio. In the final phase, intermediate data are moved dynamically at runtime to the location of the tasks that are ready for execution. To estimate runtimes instead of just considering the capacity of CPU, SABA also considers I/O, network bandwidth, memory, and storage. It also considers the data transfer cost from one datacenter to another as well as storage used for input and output. SABA did not consider the billing models imposed by different cloud providers which result in higher VM cost than expected. Authors used the Optimum monetary Cost (OC) to examine the performance of the different algorithms under various budget constraints.

OC is given by $OC = \sum_{i=1}^{N} (et_{ti} . count(t_i). p_h)$

$\mathbf{et_{ti}}$is the execution time of a task t_i, $\mathbf{count(t_i)}$ is the number of required type of physical host h for task $t_i$and $\mathbf{p_h}$ is the monetary cost per second of host h

4.3. Particle Swarm Optimization–based Resource Provisioning and Scheduling Algorithm

Rodriguez and Buyya [36] proposed static resource provisioning and scheduling strategy which minimizes the cost under a deadline constraint with meta-heuristic as an optimization strategy. This algorithm also considers the basic features of cloud computing such as the pay-as-you-go model, elasticity, dynamicity and heterogeneity of the unlimited computing resources, as well as performance variations and the boot time of VMs. Both resource provisioning and scheduling are merged as PSO problem. The proposed algorithm has an overall complexity of order O ($N * T^2 * R$) per iteration, where N is a number of particles, T is a number of tasks and R is a number of resources being used. Both resource provisioning and scheduling are combined as PSOP, the output is a near optimal schedule which determines the number and types of VMs, as well as their leasing periods and task to resource mapping. The advantage of the algorithm is generating the high-quality schedules with global optimization technique. They also introduced a performance degradation percentage that would be experienced by VMs when calculating the run times.

4.4. Multi-objective Heterogeneous earliest Finish Time

Durillo and Prodan developed the MOHEFT algorithm [37] as an extension of the classical DAG scheduling HEFT algorithm [38] for mono-objective scheduling. A Pareto-based list scheduling heuristic computes a set of tradeoff optimal solutions from which the user can select the one which suits their requirements better. MOHEFT builds several intermediate workflow solutions in parallel in each step instead of a single one as done by HEFT. MOHEFT uses dominance relationships to ensure the quality of the tradeoff solutions and a metric called crowding distance of polynomialcomplexity to guarantee their diversity. The algorithm is generic in number and type of objectives for optimizing the makespan and cost when running applications in an Amazon-based commercial Cloud. A Pareto front is an efficient tool for decision support which allows the user to select the best tradeoff solutions on their requirements. Their experiments proved that price can be reduced by half with marginal 5 % increaseof makespan. The approximate time complexity of MOHFET is O(n*m), where n and m are the number of tasks and resources respectively.

4.5. Fault-tolerant Scheduling using Spot Instances

Poola et al [39] proposed an algorithm with an objective to minimize the execution cost while meeting deadline constraint that schedules tasks on two different cloud pricing resources, spot and on-demand instances with just-in-time and adaptive scheduling heuristic. Authors define the new term LTO (Latest Time to On-Demand) which determines when the algorithm should switch to on-demandinstancestosatisfythedeadlineconstraint. At time t, LTO is the difference between the deadline and critical path, LTO_t = D - CP_t. As the run time and critical path vary depending on the instance type, authors proposed two algorithms namely Conservative and Aggressive. Conservative algorithm estimates CP and LTO on the lowest cost of the instance where as Aggressive algorithm estimates highest cost of the instance. An intelligent bidding strategy for spot VMs is proposed. The bid starts with the initial spot price and increases gradually with the progress of workflow execution and ends closer to on-demand price as execution nears the LTO. This lowers the risk of out-of-bid events as the execution nears the LTO, making sure that probability of meeting deadline constraint. This algorithm addresses problem of meeting deadlines with dynamically priced unreliable VMs under variable performance. However, the drawback of the algorithm is the cost of storing checkpoints may considerably increase the infrastructure costs.

4.6. IaaS Cloud Partial Critical Path

Saeid et al [40] proposed an algorithm, IaaS Cloud Partial Critical Paths(IC-PCP) has an objective to minimize the execution cost while satisfying the user-defined deadline constraint. The algorithm first begins by finding the partial critical path (PCP) associated with each exit node in the workflow. The tasks on each path are then scheduled on the cheapest service which can meet the latest finish time requirement of the tasks. This procedure continues recursively until all of the workflow tasks are scheduled. Along with IC-PCP, authors proposed the IC-PCPD2(IC-PCP with deadline distribution). The difference between the two algorithms is IC-PCP schedules the tasks of the partial critical path on the same VM or computation service (existing or a new) which can execute before its latest finish time. Whereas IC-PCPD2 schedule each individual task on the cheapest VM that can finish it on time. According to experimental results, IC-PCP outperforms IC-PCPD2. One of the advantages of IC-PCP is since all the tasks are scheduled on the same instance data transfer times between the tasks are zero; data transfer times will have a high impact on the makespan and execution cost of a workflow. A disadvantage of this algorithm is, it does not consider the VM startup times and resource performance variation.

4.7. Enhanced IC-P CP with Replication

Calheiros and Buyya [41] propose the EIPR algorithm which is enhanced IC-PCP with replication, provides a solution for scheduling and provisioning using idle time of provisioned VMs and a budget surplus to replicate the tasks to mitigate effects of performance variations of resources to meet the deadlines of workflow applications. The first step of EIPR algorithm solves two sub- problems namely provisioning and scheduling. The provisioning problem determines number and type of VMs to use and scheduling problem determines the order and placement of tasks on the selected resources determined during provisioning problem. This is achieved using the heuristic of IC-PCP, that is identifying and assigning of all the tasks of a partial critical path (PCP) of the workflow to the same virtual machine. The second step of the EIPR algorithm determines the start and stop times of VMs and input and output data transfer times. Finally, tasks are replicated in idle time slots of provisioned VMs or on new VMs if the replication budget slows for it. The algorithm prioritizes the replication of tasks with a ratio between execution to available time, then tasks with long execution time followed by the number of children. EIPR mitigates the effect of the poor performance of cloud resources by exploiting the elasticity and billing scheme of clouds.

4.8. Workflow Scheduling Considering 2 SLA Levels

Genez et al [42] proposed a SaaS provider offering a workflow execution service to its customers by considering two types of SLA contracts that can be used to lease VMs from IaaS providers: on-demand or reserved instances. In the proposed model, SaaS has a pool of reserved instances used to execute workflows submitted by a user before user defined deadline. On the other hand, if the infrastructure is not enough to satisfy the user deadline then on-demand resources are required to meet the user defined deadlines. They proposed scheduling problem as the integer linear programming (ILP) with the objective of minimizing the monetary cost and SLA violations. To derive a feasible schedule from the relaxed version of ILP, they proposed two heuristics namely begin-minimum end-maximum times (BMEMT) and begin-minimum time (BMT). Even though the algorithm is developed in the context of SaaS/PaaS provider potentially serving multiple customers, the solution is designed to schedule a single workflow at a time. Simulation results shows that their algorithm is capable of selecting best suited IaaS provider as well as VMs required to guarantee the QoS parameters. One of the concerns in this mode is the scalability. The number of variables and constraints increases rapidly when the number of providers and number and type of VMs that can be leased from IaaS providers and the number of tasks in the submitted workflow.

4.9. Partitioned Balanced Time Scheduling

Byun et al. [43] proposed an algorithm PBTS (Partitioned Balanced Time Scheduling), for estimating the number of VMs required to execute the workflow within a given deadline in a set of homogeneous VMs by partitioning the execution. Application running time is divided into time partitions equal to time charge unit of IaaS resource provisioning system. For each partition, PBTS first identifies the set of tasks to run then it estimates the total number of VMs required to run the tasks during the partition using balanced time scheduling algorithm [44]. Finally, tasks are executed on the allocated actual VMs on the basis of schedule obtained from running BTS. The disadvantage of this algorithm is, for finer-grained billing periods, such as 1 minute, may not be successful as tasks are unlikely to finish within a single partition and clearly works for coarse-grained billing periods such as 1 hour.

4.10. Static Provisioning Static Scheduling and Dynamic Provisioning Dynamic Scheduling

Malawski et al. [45] proposed DPDS and WA-DPDS dynamic algorithms and one static algorithm, SPSS to schedule workflow ensembles which maximizes the number of executed workflows from ensembles while meeting both budget and deadline constraints. Dynamic Provisioning Dynamic Scheduling (DPDS) algorithm consists of two phases, provisioning and scheduling. Provisioning phase calculates the initial number of VMs to use on the basis of available budget and time. On the basis of VM utilization, VM pool is updated periodically. If the utilization falls below the predefined thresholds, then VMs are shutdown. And if utilization is above the threshold and budget allows for it then new VMs are leased. In the scheduling phase, ready tasks from workflow ensemble are assigned to the priority queue. If the priority queue is not empty and idle VMs are available, then the task from the priority queue is submitted to arbitrary idle VM. This step is repeated until the priority queue is empty or no idle VMs available. The Workflow-Aware DPDS (WA-DPDS) algorithm is variant of a DPDS which incorporates an admission control procedure which accepts new workflow for execution only when there is enough budget is available. Otherwise, workflow is rejected and tasks are removed from the queue. Static Provisioning Static Scheduling (SPSS) assigns sub-deadlines to each task based on the slack time of the workflow (amount of extra time that a workflow can extend its critical path and still be completed by the ensemble deadline). If there are no time slots, then new VMs are leased to schedule the tasks. Simulation results show that static algorithm outperforms dynamic algorithms.

4.11. SPSS-ED and SPSS-EB

Pietri et al. [46] proposed two energy-aware algorithms SPSS-ED and SPSS-EB for resource provisioning to schedule workflow ensembles on basis of SPSS. The first algorithm called SPSS-ED focuses on meeting energy and deadline constraints while another one called SPSS-EB focuses on meeting energy and budget constraints. The Aim of both SPSS-ED and SPSS-EB is to maximize the number of executed workflow minimizing energy consumption. For each workflow in the ensemble, SPSS-EB plans the execution of workflow by scheduling each task. It accepts the plan only on meeting energy and budget constraints. The Same process is used in SPSS-ED, instead of budget, deadline constraint is used. This work does not consider the different workflow structures, data transfer costs, provisioning and deprovisioning delays.

4.12. Dyna

Zhou et al. [47] proposed a framework work for scheduling system called Dyna by considering the dynamic nature of cloud environment and price dynamics like spot and on-demand instances. The aim of the Dyna is to minimize the monetary cost while meeting the probabilistic deadline that reflects the performance variability of the resources and the price dynamics of spot instances. Spot instances are used to reduce the infrastructure costs and on-demand instance to meet the deadline constraint by generating the hybrid instance configuration plan. In this plan, the task is initially assigned to a spot instance. If the task fails in this instance, it will be reassigned to another instance of the next type in the configuration plan until the task is executed successfully, since last instance type in on-demand instance task can always finish the execution.

Table 5 shows the comparison of the algorithms.

Table 5. The comparison of the algorithms

Algorithm	Workflow Dynamicity	Scheduling Objective	Optimization Strategy	Data Transfer Cost	Storage Cost	Task Resource mapping
Malawski et al [34]	Single	Deadline and cost	Hybrid HO	Yes	No	Single
SABA [35]	Single	Budget, makespan and security	Heuristic	Yes	Yes	Single
Rodriguez &Buyya [36]	Single	Deadline and cost	Meta-Heuristic	No	No	Single
MOHEFT [37]	Single	Generic multi objective	Heuristic	Yes	Yes	Single
Poola et al [39]	Single	Deadline, cost and reliability	Heuristic	No	No	Single
IC-PCP/IC-PCPD2 [40]	Single	Deadline and cost	Heuristic	No	No	Single
EIPR [41]	Single	Deadline and cost	Heuristic	No	No	Single
Genez et al [42]	Single	Deadline and cost	Optimal	No	No	Single
PBTS [43]	Single	Deadline and cost	Heuristic	No	No	Multiple
SPSS [45]	Ensemble	Budget, deadline and workload	Heuristic	No	No	Single
DPDS [45]	Ensemble	Budget, deadline and workload	Heuristic	No	No	Single

Algorithm	Workflow Dynamicity	Scheduling Objective	Optimization Strategy	Data Transfer Cost	Storage Cost	Task Resource mapping
SPSS-ED [46]	Ensemble	Deadline, workload and energy	Heuristic	No	No	Single
SPDD-EB [46]	Multiple	Budget, workload and energy	Heuristic	No	No	Single
Dyna [47]	Multiple	Deadline and cost	Heuristic	No	No	Single
SCS [48]	Multiple	Deadline and cost	Heuristic	No	No	Single
WPRS [49]	Single	Deadline and cost	Hybrid HO	No	No	Single
RNPSO [50]	Single	Deadline and cost	Meta-Heuristic	No	No	Single
Zhu et al [51]	Ensemble	Makespan and cost	Meta-Heuristic	No	No	Single

4.13. Scaling-Consolidation-Scheduling (SCS)

Mao and Humphrey proposed [48] is a deadline constrained algorithm that has auto scaling mechanism to dynamically allocating/deallocating VMs on the basis of the current status of the tasks. In the first step, tasks are bundled which prefer the same type of instance and have one-to-one task dependencies which also reduces the data transfer times by distributing the overall deadline among the tasks. Then a load vector is created by determining the most efficient VM type for each task. Load vectors are updated for each scheduling cycle and indicate how many machines of each type are required to complete the execution of tasks to meet deadline constraint with minimum cost. Afterward all the partial instance hours are consolidated by merging the tasks running on different virtual machines types into a single one. Then tasks are scheduled on each VM using Earliest Deadline First (EDF) algorithm.

5. RESEARCH CHALLENGES

Scheduling algorithms need to address various challenges derived from the cloud resource model. In this section, we will discuss the challenges in workflow scheduling technology in the cloud.

1) Most of the workflow scheduling algorithms focus on the budget and deadline of the workflow, ignoring other factors such as resource utilization, reliability, acquisition/termination delay and fault- tolerance. On-demand access to ‘unlimited’ resources, lead to unnecessary cost by not considering the IaaS model, we should pay attention to the improvement of resource utilization too.
2) Several studies demonstrated the importance of addressing resource provisioning problem, and proved that resource provisioning strategy affects cost and makespan. Choosing the optimal configuration of the VMs to run the tasks is a challenging problem. If the resources are overprovisioned, then the system utilization will be low. On the other hand, if resources are underprovisioned, the scheduler will not be able to achieve the expected performance. Researchers need to focus on the below challenges in resource provisioning
   a) Choose the right optimal configuration of VMs
   b) Dynamically scale in and out the resource pool
   c) VM provisioning and deprovisioning delays
   d) Dynamic pricing scheme of resources
3) Other, time overhead in scientific workflows in the cloud computing environment is data transfer cost. Datasets are usually big in scientific workflows. The data transfer time for these big data sets across different data centers can be enormous. Researchers need to focus data transfer and data placement strategies.
4) Most of the workflow scheduling algorithms are implemented in cloud simulators. We must apply cloud scheduling algorithms to the real cloud so that we can solve more real practical problems.
5) One of the major concerns of the cloud providers is the Power consumption. New techniques/approaches should be developed for an energy efficient resource management system to minimize the total power consumption of the cloud data centers.

6. CONCLUSION

This paper describes the workflow scheduling techniques by considering the applications modeled as DAGs and the resources offered by different cloud providers. We started with definition and model of

DAG, followed by basic details of scientific workflows. In this paper, we have given an introduction and overview of existing scheduling techniques and comprehensive study of some of the outstanding algorithms.

REFERENCES

[1] Wu, F., Wu, Q., Tan, Y., Wang, W., & Sun, X., "Schedule Compaction and Deadline Constrained DAG Scheduling for IaaS Cloud," *Cloud Computing and Big Data Lecture Notes in Computer Science*, pp. 16-28, 2015.

[2] A. Kumar, "World of Cloud Computing & Security," *International Journal of Cloud Computing and Services Science (IJ-CLOSER)*, vol. 1, no. 2, pp. 53-58, 2012.

[3] Ludäscher, B., Weske, M., Mcphillips, T., & Bowers, S., "Scientific Workflows: Business as Usual," *Lecture Notes in Computer Science Business Process Management*, pp. 31-47, 2009.

[4] Zhiyuan Shi, Pingbo Chen, Lianfen Huang. "A New Efficient Dynamic System Information Scheduling Strategy in TDD-LTE," *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, vol. 11, no. 9, pp. 5480- 5489, 2013.

[5] Kwok, Y., Ahmad, I., "Static scheduling algorithms for allocating directed task graphs to multiprocessors," *ACM Computing Surveys*, vol. 31, no. 4, pp. 406-471, 1999.

[6] Sirisha Potluri, Katta Subba Rao, "Quality of Service based Task Scheduling Algorithms in Cloud Computing," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 7, no. 2, pp. 1088-1095, 2017.

[7] Yu, J., Buyya, R., &Ramamohanarao, K., "Workflow Scheduling Algorithms for Grid Computing," *Studies in Computational Intelligence Metaheuristics for Scheduling in Distributed Computing Environments*, pp. 173-214.

[8] Maheswaran, M., Ali, S., Siegal, H., Hensgen, D., & Freund, R., *"Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems,"* Proceedings Eighth Heterogeneous Computing Workshop (HCW'99).

[9] Topcuoglu, H., Hariri, S., & Wu, M., "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, no. 3, pp. 260-274, 2002.

[10] Sakellariou, R., Zhao, H., *"A hybrid heuristic for DAG scheduling on heterogeneous systems,"* 18th International Parallel and Distributed Processing Symposium, 2004. Proceedings.

[11] Xiao, P., Hu, Z., Zhang, Y., "An Energy-Aware Heuristic Scheduling for Data-Intensive Workflows in Virtualized Datacenters," *Journal of Computer Science and Technology*, vol. 28, no. 6, pp. 948-961, 2013.

[12] Yang, T., Gerasoulis, A., *"A fast-static scheduling algorithm for DAGs on an unbounded number of processors,"* Proceedings of the 1991 ACM/IEEE conference on Supercomputing - Supercomputing '91.

[13] Kim SJ, Browne JC, *"A general approach to mapping of parallel computation upon multiprocessor architectures,"* In*: Proceedings of the 1991 ACM/IEEE conference on supercomputing '91, ACM/IEEE,* pp. 633-642.

[14] Sarkar, V. "Partitioning and scheduling parallel programs for multiprocessors," *London, England: Pitman Pub*.

[15] Liou, J., Palis, M. A., & Wei, D. S. "Performance Analysis of Task Clustering Heuristics for Scheduling Static Dags On Multiprocessor System," *Parallel Algorithms and Applications*, vol. 12, no. 1-3, pp. 185-203.

[16] Darbha, S., Agrawal, D., "Optimal scheduling algorithm for distributed-memory machines," *IEEE Transactions on Parallel and Distributed Systems*, vol. 9, no. 1, pp. 87-95, 1998.

[17] Colin, J. Y., and Chrétienne, P., "C. P. M. Scheduling with Small Communication Delays and Task Duplication," *Operations Research*, vol. 39, no. 4, pp. 680-684, 1991.

[18] Palis, M., Lieu, J., Wei, D., "Task clustering and scheduling for distributed memory parallel architectures," *IEEE Transactions on Parallel and Distributed Systems*, vol. 7, no. 1, pp. 46-55, 1996.

[19] Park, G., Shirazi, B., & Marquis, J, *"DFRN: a new approach for duplication based scheduling for distributed memory multiprocessor systems,"* Proceedings 11th International Parallel Processing Symposium.

[20] Kruatrachue, B., Lewis, T, "Grain size determination for parallel processing," *IEEE Software*, vol. 5, no. 1, pp. 23-32,1988.

[21] Chung, Y., Ranka, S, *"Applications and performance analysis of a compile-time optimization approach for list scheduling algorithms on distributed memory multiprocessors,"* Proceedings Supercomputing '92.

[22] Chen, H., Shirazi, B., Marquis, J. *"Performance evaluation of a novel scheduling method: Linear clustering with task duplication,"* In Proceedings of the 2nd International Conference on Parallel and Distributed Systems, pp. 270–275, Dec.1993.

[23] Ahmad, I., Kwok, Y, "On exploiting task duplication in parallel program scheduling," *IEEE Transactions on Parallel and Distributed Systems*, vol. 9, no. 9, pp. 872-892, 1988.

[24] Papadimitriou, C., Yannakakis, M, *"Towards an architecture-independent analysis of parallel algorithms,"* Proceedings of the twentieth annual ACM symposium on Theory of computing - STOC '88.

[25] Guodong, L., Daoxu, C., Daming, W., Defu, Z, *"Task clustering and scheduling to multiprocessors with duplication,"* Proceedings International Parallel and Distributed Processing Symposium.
[26] Hartmanis, J, "Computers and Intractability: A Guide to the Theory of NP-Completeness (Michael R. Garey and David S. Johnson)," *SIAM Review*, vol. 24, no. 1, pp. 90-91.
[27] Blythe, J., Jain, S., Deelman, E., Gil, Y., Vahi, K., Mandal, A., Kennedy, K, *"Task scheduling strategies for workflow-based applications in grids,"* CCGrid 2005, IEEE International Symposium on Cluster Computing and the Grid, 2005.
[28] Yarkhan, A., Dongarra, J. J, "Experiments with Scheduling Using Simulated Annealing in a Grid Environment," Grid Computing — GRID 2002 Lecture Notes in Computer Science, pp. 232-242.
[29] Young L, McGough S, Newhouse S, Darlington J, *"Scheduling Architecture and Algorithms within the ICENI Grid Middleware,"* In: Proceedings of UK e-science all hands meeting, Citeseer, pp. 5-12.
[30] Sonmez, O., Yigitbasi, N., Abrishami, S., Iosup, A., Epema, D., *"Performance analysis of dynamic workflow scheduling in multicluster grids,"* Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing - HPDC '10.
[31] Menasce, D., &Caslicchio, E, *"A framework for resource allocation in grid computing,"* The IEEE Computer Society's 12th Annual International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems, 2004, (MASCOTS 2004), Proceedings.
[32] Yu, J., Buyya, R., Tham, C. K, *"Cost-Based Scheduling of Scientific Workflow Application on Utility Grids,"* First International Conference on e-Science and Grid Computing (e-Science'05).
[33] Sakellariou, R., Zhao, H., Tsiakkouri, E., Dikaiakos, M. D, "Scheduling Workflows with Budget Constraints," *Integrated Research in GRID Computing*, pp. 189-202.
[34] Malawski, M., Figiela, K., Bubak, M., Deelman, E., &Nabrzyski, J., "Scheduling Multilevel Deadline-Constrained Scientific Workflows on Clouds Based on Cost Optimization," *Scientific Programming*, pp. 1-13, 2015.
[35] Zeng, L., Veeravalli, B., Li, X., "SABA: A security-aware and budget-aware workflow scheduling strategy in clouds," *Journal of Parallel and Distributed Computing*, 75, pp. 141-151, 2015.
[36] Rodriguez, M. A., Buyya, R., "Deadline Based Resource Provisioningand Scheduling Algorithm for Scientific Workflows on Clouds," *IEEE Transactions on Cloud Computing*, vol. 2, no. 2, pp. 222-235, 2014.
[37] Durillo, J. J., Prodan, R. "Multi-objective workflow scheduling in Amazon EC2," *Cluster Computing*, vol. 17, no. 2, pp. 169-189, 2013.
[38] Topcuoglu, H., Hariri, S., Wu, M. "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, no. 3, pp. 260-274, 2002.
[39] Poola, D., Ramamohanarao, K., Buyya, R, "Fault-tolerant Workflow Scheduling using Spot Instances on Clouds," *Procedia Computer Science*, 29, pp. 523-533, 2014.
[40] Abrishami, S., Naghibzadeh, M., Epema, D. H., "Deadline-constrained workflow scheduling algorithms for Infrastructure as a Service Clouds," *Future Generation Computer Systems*, vol. 29, no. 1, pp. 158-169, 2013.
[41] Calheiros, R. N., Buyya, R., "Meeting Deadlines of Scientific Workflows in Public Clouds with Tasks Replication," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 7, pp. 1787-1796, 2014.
[42] Genez, T. A., Bittencourt, L. F., Madeira, E. R, "Workflow scheduling for SaaS / PaaS cloud providers considering two SLA levels," *IEEE Network Operations and Management Symposium*, 2012.
[43] Byun, E., Kee, Y., Kim, J., Maeng, S, "Cost optimized provisioning of elastic resources for application workflows," *Future Generation Computer Systems*, vol. 27, no. 8, pp. 1011-1026, 2011.
[44] Byun, E., Kee, Y., Kim, J., Deelman, E., Maeng, S, "BTS: Resource capacity estimate for time-targeted science workflows*," Journal of Parallel and Distributed Computing*, vol. 71, no. 6, pp. 848-862.
[45] Malawski, M., Juve, G., Deelman, E., Nabrzyski, J, *"Cost- and deadline-constrained provisioning for scientific workflow ensembles in IaaS clouds,"* 2012 International Conference for High Performance Computing, Networking, Storage and Analysis.
[46] Pietri, I., Malawski, M., Juve, G., Deelman, E., Nabrzyski, J., Sakellariou, R., *"Energy-Constrained Provisioning for Scientific Workflow Ensembles,"* 2013 International Conference on Cloud and Green Computing.
[47] Zhou, A. C., He, B., Liu, C., "Monetary Cost Optimizations for Hosting Workflow-as-a-Service in IaaS Clouds," *IEEE Transactions on Cloud Computing*, vol. 4, no. 1, pp. 34-48, 2016.
[48] Mao, M., Humphrey, M, *"Auto-scaling to minimize cost and meet application deadlines in cloud workflows,"* Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis on - SC '11.
[49] Rodriguez, M. A., Buyya, R, *"A Responsive Knapsack-Based Algorithm for Resource Provisioning and Scheduling of Scientific Workflows in Clouds,"* 44th International Conference on Parallel Processing, 2015.
[50] Li, H., Fu, Y., Zhan, Z., Li, J., *"Renumber strategy enhanced particle swarm optimization for cloud computing resource scheduling"*, IEEE Congress on Evolutionary Computation (CEC), 2015.

[51] Zhu, Z., Zhang, G., Li, M., Liu, X, "Evolutionary Multi-Objective Workflow Scheduling in Cloud," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 5, pp. 1344-1357, 2016.