Workflow Scheduling on the Hybrid Cloud to Maintain Data Privacy under Deadline Constraint

Hamid Abrishami, Amin Rezaeian, Mahmoud Naghibzadeh
Dept of Computer Engineering
Ferdowsi University of Mashhad, Iran
ha.abrishami@stu.um.ac.ir, amin.rezaeian@stu.um.ac.ir, naghibzadeh@um.ac.ir

ABSTRACT: *The development of cloud computing technology has been continuously growing since its invention and has attracted the attention of many researchers in the academia and the industry, particularly during the recent years. The majority of organizations, whether large corporate businesses or typical small companies, are moving towards adapting this cutting edge technology. The private cloud provides low cost and privacy for workflow applications execution. However, an organization's requirements to high performance resources and high capacity storage devices encourage them to utilize public clouds. Public cloud leases information technology services in the form of small units and in larger scale compared to private cloud, but this model is potentially exposed to the risk of data and computation breach and is less secure in comparison to a pure private cloud environment. The combination of public and private clouds is known as hybrid cloud, where workflow tasks can be executed on resources residing on either public or private clouds. The objective of this paper is to present a scheduling algorithm for maintaining data privacy in workflow applications, such that the budget is minimized, while the makespan limitation imposed by the user is satisfied. A scheduling algorithm called Workflow scheduling on Hybrid Cloud to maintain Data Privacy (WHPD) is proposed and it is shown that it can perform as well as HCOC while preserving data and computation privacy.*

Keywords: Hybrid Cloud, Scheduling Algorithm, Static Scheduling, Workflow, DAG, Sensitive Tasks

Received: 18 June 2015, Revised 26 July 2015, Accepted 30 July 2015

1. Introduction

In the cloud computing, services can be provided to users in dynamic and scalable methods and with different payment models. These facilities can be delivered using hardware and software virtualization [1]. Cloud services are presented on users' demand and every user can only access her requested resources. This would lower the cost compared to having one's own and maintain expensive facilities and would allow users and specially small and medium companies to use cutting-edge software and platforms. It would additionally provide them with better management for distributed resources, enabling them to compete with large enterprises [2].

However, one of the most important challenges academic and industrial organizations are facing is dealing with security issues in cloud computing [3], [4]. As per an investigation conducted by the International Data Corporation (IDC), security and privacy are known as the most important challenges amongst nine existing threats for cloud [5]. This research indicates that it is not acceptable to transfer sensitive data to cloud's computing resources over the network, and organizations prefer to use a model

called hybrid cloud [6].

In hybrid cloud models, sensitive information and processes of the organization be kept in private clouds. A private cloud is a relatively small infrastructure which is installed inside the organization. The other parts of the information and computation that are not as sensitive, can be moved to a public cloud on demand [4], [7].

Gartner Inc., a technology research and advisory company, has recently announced that until 2016, the majority of expenses involved in information theory will be dedicated to preparing infrastructures for cloud computing. This organization predicts that more than half of enterprises will use hybrid cloud in 2017 [8].

Complicated applications are usually presented as workflows. Directed Acyclic Graphs (DAG) are conventional models to present workflows, where nodes are tasks with computation cost (e.g. number of instructions), and edges are communications between tasks, which are labeled with communication costs (e.g. volume of data to be transferred in bytes). A sample workflow is shown in Figure 1.

Tasks in a workflow must be mapped to computational resources with regards to some optimization criteria. This mapping is called workflow scheduling problem, which is proven to be NP-Complete [9], and therefore there is no known algorithm with polynomial time complexity that can find an optimal solution for this problem.

In this paper, we focus on solving the problem of scheduling workflow applications on hybrid clouds. To this aim, we propose a static heuristic algorithm which schedules sensitive tasks on private clouds and uses public cloud's resources to meet user's deadlines. The proposed algorithm further tries to minimize the cost of using public clouds.

The rest of this paper is organized as follows. The next section gives an overview about hybrid cloud scheduling methods which have been proposed in the literature. In Section 3, we introduce the problem formulation and system model for hybrid cloud. Our proposed method is presented in Section 4. Further, experimental results are discussed in Section 5 and eventually we conclude the paper in Section 6.

2. Previous Works

Characteristics of hybrid cloud computing such as maintaining security of resources, dynamic infrastructure scaling and pay-per-use model, must be considered in scheduling workflows, and therefore new scheduling methods are required that take these features into account.

Scheduling algorithms that are designed to cover hybrid cloud characteristics must be aware of the scalability of the service environment they reside in, and they must try to decrease cost while maintaining service efficiency. Enforcing these features in the scheduling algorithms convince the organizations that not only connecting to resources in public cloud is secure, but it's cost-effective.

In order to schedule workflows on cloud computing systems, QoS-based algorithms must be used. These algorithms try to optimize multiple criteria at the same time. They include heuristic methods, meta-heuristic-methods, and algorithms based on mathematical modeling [10]. Mathematical methods use mathematical optimization approaches like linear programming or game-theory to solve scheduling problems. Meta-heuristics are also popular in solving multi-criteria problems, however these methods need lots of iterations to find a desirable answer which makes them too time-consuming.

Rahman et al. [11] proposed a "*Hybrid Heuristic for Scheduling Data Analytics Workflow*" (AHH) algorithm to solve scheduling problems. This algorithm uses genetic algorithm to solve the problem of scheduling workflows on hybrid clouds. This algorithm distributes deadline and budget among tasks using a genetic algorithm. Then using Dynamic Critical Path (DCP) [12], the algorithm assigns ready tasks level by level, considering sub-deadlines and sub-budgets. This algorithm is able to optimize multiple criteria for scheduling.

The latest group of QoS-based scheduling methods are heuristic algorithms. These algorithms are usually fast, however they do

not guarantee the quality of the solution. Since quality is not guaranteed, these methods can potentially be further improved. Bittencourt and Madeira [13] proposed "Hybrid Cloud Optimized Cost scheduling" (HCOC) to schedule workflows on hybrid cloud platforms composed of multi-cores. This method accepts the deadline, D_G, as input and tries to schedule the corresponding workflow with minimum cost. It starts using only the resources on the private cloud, and while the scheduling solution violates the user defined deadline, new resources are leased from the public cloud and new tasks are candidate to run on them, then the algorithm reschedules the workflow. This method uses Path Clustering Heuristic (PCH) approach [14], which is a popular scheduling algorithm on the utility grids, as a foundation for the scheduling. This method shows best performance when the infrastructure includes multi-core processors. However, the rescheduling algorithm that HCOC uses for the candidate tasks is HEFT [15], which shows lower performance than PCH for consistent heterogeneous platforms.

Malawski et al. [16] proposed a static provisioning static scheduling algorithm (SPSS) on the public cloud which is based on the vertical clustering. This method consists of two steps of deadline distribution and the scheduling step which is assigning resources to the tasks of the workflow. In the deadline distribution step, firstly the fastest schedule for the critical path is found.

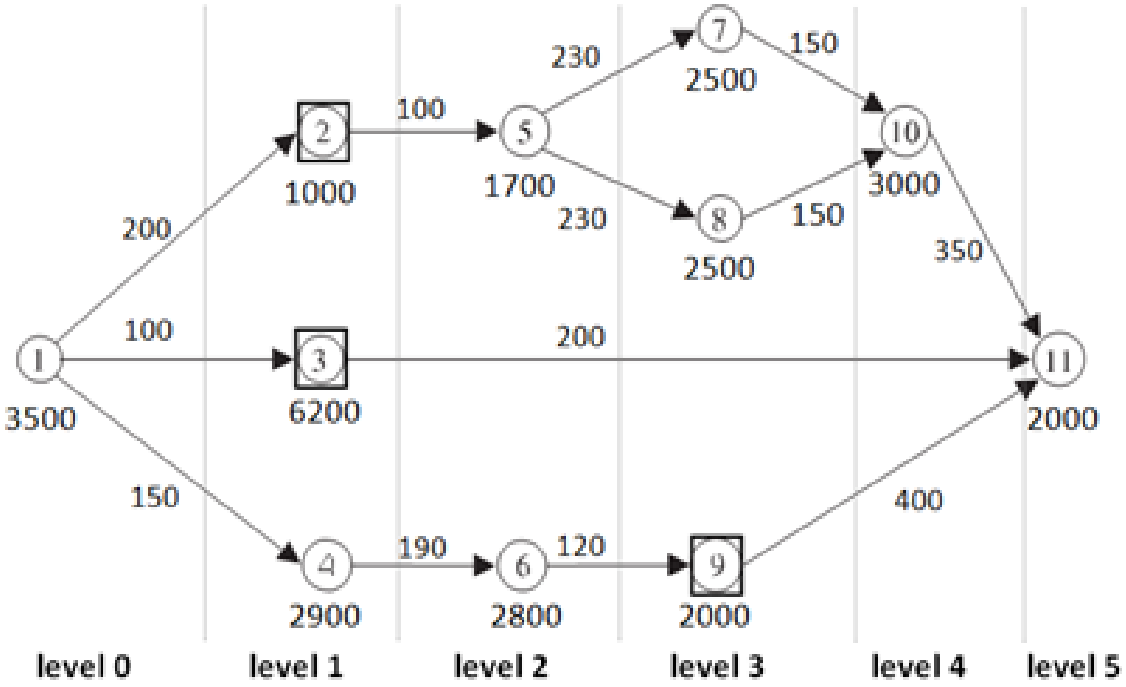


Figure 1. An example of a workflow DAG

Then, the difference between execution time of the critical path and the user defined deadline is calculated and distributed among the levels of the graph. These level sub-deadlines are used in order to calculate the sub-deadlines of tasks in the graph. In the scheduling step, each task is assigned to the cheapest public cloud resource which is able to finish it before its sub-deadline.

Chopra and Singh [17] presented a level-based scheduling algorithm in order to schedule workflow applications on hybrid cloud. In this algorithm, all tasks in each level are executed in parallel, and once all tasks in the current level are executed, the resources will be released and the execution of the next level will start. Before allocating resources to tasks in a given level, the rank of every task in that level is calculated. Ranks are measured based on tasks' runtimes and communications. In addition, a sub-deadline is estimated for every task and then whether the task's runtime meets its sub-deadline or not, a resource on private or public clouds is assigned to it. Since this method tries to minimize the cost, if a resource on private cloud meets the task's sub-deadline, it is allocated to that task; otherwise it is allocated to an appropriate resource on the public cloud. In order to estimate sub-deadline for a task, this method uses the ratio of weight of that task to the sum of all weights in the workflow. Several methods are proposed in order to facilitate security and privacy for transferring information. These methods consist of cryptography, using Virtual Private Networks (VPN) and Scheduling [18].

3. System Model

As illustrated in Fig.1, workflows can be presented using Directed Acyclic Graphs (DAG). In this model, every independent task and task dependencies are shown as nodes and edges, respectively.

Existence of an edge between two nodes indicates that execution of the succeeding node cannot be started before the execution of its proceedings are finished, and all required data has been received from the preceding nodes. Costs of computation and communication are respectively presented using labels on nodes and edges. This graph is defined by $G = (V, E)$, and it has n nodes (representing n tasks), in which $t_a \in V$ is a task of the workflow. $instructions_{ta} \in R^+$ is computational cost, related to task t_a. Besides, $e_{a,b} \in E$ (where $a \neq b$) specifies the dependency between tasks t_a and t_b. Communication cost of edge $e_{a,b}$ is denoted by $data_{a,b} \in R^+$. In a workflow, tasks with no parent are called entry tasks. These kinds of tasks are denoted by t_{entry}. Tasks with no children are denoted with t_{exit} and are called exit tasks. If there are more than one t_{entry}, a single t_{entry} with $w = 0$ is added to the graph and its dependencies to previous t_{entry} nodes will be established. A similar process occurs if there are more than one t_{exit} nodes.

Set of sensitive tasks that are related to sensitive computational resources, are represented using $\hat{A} \subseteq V$. This set is given by user. For instance, in Figure 1, tasks that are illustrated with a square are given as sensitive tasks.

A private cloud is a set of heterogeneous resources denoted with $R = \{r_1, r_2, \ldots, r_k\}$. Each resource r_i has the computational capacity of $P_{ri} \in R^+$. Resources are connected through a network. In addition to resources in the private cloud, there are resources available in the public cloud. These resources are virtualized and costly, which are defined by $U = \{u_1, u_2, \ldots, u_w\}$. These resources are considered to be unlimited; however they are used on demand. Computational capacities of these resources are denoted by $P_{uj} \in R^+$.

In addition to computational capacity, each resource on the public cloud has a price, which is given by "cost in time unit". We define this as $price_{uj}$ for resource u_j. These prices are considered zero for the private cloud.

A hybrid cloud is a set of resources, consisting of resources on the private cloud and leased resources of the public cloud. Resources from the public cloud is defined by $S_u \subseteq U$. Equation 1 defines H as set of the hybrid cloud resources.

$$H = \left(\bigcup_{a=1}^{b} R_a \right) \cup \left(\bigcup_{s=1}^{t} S_s \right) \tag{1}$$

Where $b \geq 1$ and $t \geq 1$.

In this model, resources are connected through a network, and the minimum bandwidth between resources r_i and r_j, which could be either on the public or private cloud, is defined as $l_{ri, rj}$. Equation 2 formulates $w_{i, r}$ that is the runtime for task i on resource r.

$$w_{i,r} = instructio\ ns_i / p_r \tag{2}$$

In this equation, p_r specifies the computational capacity of resource r. In order to compute data transfer time between tasks i and j that are assigned to resources r and p respectively, Equation 3 is used.

$$c_{i,j} = data_{i,j} / l_{r,p} \tag{3}$$

The cost of running task t_a is calculated by multiplying runtime of the task ($w_{i, r}$) by price of the assigned resource ($price_r$).

The total execution time of a workflow is denoted by *makespan*, and is defined by the timespan between start time of the entry task (t_{entry}) and finish time of the exit task (t_{exit}). It is calculated in Equation 4.

$$makespan = max\{AFT(t_{exit})\} - min\{AFT(t_{entry})\} \tag{4}$$

In this equation, $AFT(t_{exit})$ and $AST(t_{entry})$ specify actual finish time for t_{exit} and actual start time for t_{entry} respectively.

In order to find the sub-deadline of a given task, we use its level in the graph. The level of a task is defined by its distance from the entry node. A sample level assignment is depicted in Figure 1. To compute the level of each task, denoted by $level(t_i)$, we use

$$level(t_i) = \begin{cases} 0 & t_i = t_{entry} \\ 1 + \max_{t_p \in parents\ of\ (t_i)} (level(t_p)) & otherwise \end{cases} \quad (5)$$

The aim of this paper, is to solve the scheduling problem, which is assigning resources to tasks, such that cost is minimized and *makespan* is less than or equal to user defined deadline. In addition, this algorithm must guarantee running of sensitive tasks on the private cloud.

4. Proposed Method

In this section, we propose a workflow scheduling method, Workflow scheduling on Hybrid Cloud to maintain Data Privacy (WHPD), on the hybrid cloud infrastructure to maintain data privacy under deadline constraint. WHPD method consists of three main steps which are deadline distribution for each level, fastest schedule and adjusting schedule. In the first step, user defined deadline is distributed between levels of the workflow graph. After this process, if tasks' runtimes do not exceed their levels' sub-deadlines, it is guaranteed that the workflow's *makespan* does not exceed the user defined deadline.

In the second step, tasks are scheduled in the order of their priority. In this step, sensitive tasks are scheduled on earliest resources on the private cloud, and other tasks are scheduled on earliest resources on the public cloud. The earliest resource for a task is the resource which finishes that task earlier than other resources. Therefore this step is very similar to HEFT [15] algorithm, however there is an additional condition to assign a task. Since this step selects earliest resources from the public cloud, the solution made by this step is very expensive and its *makespan* is usually much shorter than the deadline.

The third step tries to refine the schedule map that is produced in the previous step. To do this, the algorithm reschedules tasks that were previously scheduled on the public cloud, and moves them to free resources on the private cloud, or cheaper resources on public cloud. On each reschedule, the algorithm checks if total *makespan* is still less than or equal to user defined deadline. This step tries to minimize the cost, while maintaining the deadline condition.

In order to schedule a workflow, some primary properties for its graph must be calculated. For each unscheduled task t_i, $EST(t_i)$ is defined as its earliest start time. This is computed using Equation 6.

$$EST(t_i) = \begin{cases} 0 & t_i = t_{entry} \\ \max_{t_p \in parents\ of\ (t_i)} (EST(t_p) + w_{p,r} + c_{i,p}) & otherwise \end{cases} \quad (6)$$

In this equation $w_{p,r}$ specifies runtime of task t_p on fastest resource r, therefore earliest finish time of task t_i is denoted as $EFT(t_i)$ and it is calculated using Equation 7.

$$EFT(t_i) = EST(t_i) + w_{i,r} \quad (7)$$

Priority of a task is defined by Topcuoglu et al [15]. It is denoted by $rank(t_i)$ and is calculated using Equation 8.

$$rank(t_i) = \begin{cases} w_{i,r} & t_i = t_{exit} \\ w_{i,r} + \max_{t_p \in childs\ of\ (t_i)} (rank(t_p) + c_{i,p}) & otherwise \end{cases} \quad (8)$$

Since tasks' runtimes are unknown until they are assigned to a resource, values of $w_{i,r}$ values are calculated considering that the task is assigned to fastest resource on public cloud.

Algorithm 1 demonstrates the pseudo-code of the proposed method to schedule workflows on a hybrid cloud. In order to be more readable, all variables are considered global. User inputs are *G* as workflow graph, *D* as user defined deadline and *A* as normal tasks (that are not sensitive).

In this algorithm and in Lines 1 to 3, primary properties of the workflow are calculated. The first step of the algorithm that is

deadline distribution is called in Line 4. It will be described later in Algorithm 2, though it must be mentioned that this step is done considering only resources on the private cloud.

In Lines 5 to 7, Algorithm 1 checks if the whole graph can be scheduled on private cloud, and if it could, the schedule map is returned and the scheduling process is done.

In Line 8, sub-deadlines are calculated for each level of the graph. The second step of the proposed method that is fastest schedule is done on Line 9. This is a modified version of HEFT algorithm that considers the existence of sensitive tasks and the private cloud. The solution presented by this step is the fastest and the most expensive solution, which will be refined later (in the third step).

```
Require: DAG G, Deadline D, normal tasks A
1   Add t_entry, t_exit and their corresponding dependencies to G ;
2   Compute rank (t_i) for each task in G according to Equation 8;
3   Determine level (t_i) for each task in G according to Equation 5;
4   Call DeadlineDistribution();
5   if makespan < D then
6      Return (ScheduleMap);
7   end
8   Determine d (t_i) for each task in G according to Equation 9;
9   Call FastestSchedule ();
10  if makespan > D then
11     Return (null);
12  end
13  Call AdjustSchedule ();
14  Return (ScheduleMap);
```

Algorithm 1. The proposed scheduling algorithm

Since the second step provides the fastest schedule, in Lines 10 to 12 it is checked if the *makespan* is beyond the deadline, in which case, we cannot find a faster schedule. Therefore *null* is returned as algorithm failed to find a schedule that meets the user defined deadline. In Line 13, the third step of the proposed algorithm is called. This step takes as input the fastest schedule map, and tries to refine it. The refinement that adjusts the schedule includes rescheduling some of the tasks. This rescheduling results in selecting more cost effective resources and it decreases in cost.

Algorithm 2 shows the pseudo-code for deadline distribution that is the first step of the proposed algorithm. This routine tries to make the maximum use of capacity of resources on the private cloud. In order to use the maximum capacity of resources, we use a variable named *barrier*. It starts from zero, and stores the largest *makespan* that the resources on the private cloud have used.

Then at each iteration, Algorithm 2 tries to schedule as many ready tasks as possible before the *barrier*. Ready tasks are tasks whose parents have been scheduled. If scheduling the next ready task before *barrier* is not possible, then a ready task with the highest priority is selected and scheduled on the earliest resource.

The function Schedule WithMinEFT(R, t_i), that is called in Line 7, finds the earliest resource among resources in set R and assigns task t_i to it. This function returns the earliest finish time of the task t_i on a found resource. The *barrier* is then increased with regards to the returned earliest finish time, and this process is continued until all tasks are scheduled. The following describes Algorithm 2 in more details.

```
1   Sort all tasks in a list by nonincreasing order of rank value;
2   barrier ← 0;
3   Mark t_entry scheduled parent;
4   while list is not empty do
5     X ← tasks in list such that all parents are scheduled;
6    for all x_i ∈ X do
7      EFT(x_i , r_j) ← ScheduleWithMinEFT(R , x_i);
8      if EFT(x_i , r_j) <= barrier then
9         Assign task x_i to processor r_j;
10        Remove x_i from the planning list; Update X;
11     end
12   end
13   x_k ← first task in the X;
14   EFT(x_k , r_j) ← ScheduleWithMinEFT(R , x_k);
15   Assign task x_k to processor r_j; Remove x_k from the list;
16   barrier ← EFT(x_k , r_j);
17  end
```

Algorithm 2. Deadline distribution algorithm

At the beginning of Algorithm 2 (Lines 1 to 3), all given tasks are stored in a variable named *list*, sorted in descending order, based on their ranks. It is also considered that the task t_{entry} is ready to schedule (it is marked as its parent is scheduled) and the initial value of *barrier* is set to zero.

In Line 5, ready tasks are assigned to set *X*. In Lines 6 to 12, tasks are selected one by one from set *X* and if it can be scheduled earlier than the *barrier*, it is added to schedule map. Scheduled tasks are removed from the *list* and *X* is updated in Line 10. It means that we add new ready tasks to set *X*. These new tasks, which are children of the scheduled task, are now ready. The loop between Lines 6 to 12 iterates while there are tasks in set *X* that can be scheduled before the *barrier*.

In Line 13, the loop is finished, thus all tasks in *X* have finish times later than the *barrier*. In Lines 13 to 15, the task with the highest priority is selected from *X* and is scheduled on the earliest resource.

In Line 16, the *barrier* is updated, and the loop between Lines 4 to 17 iterates until all tasks in the *list* are scheduled. After this process, sub-deadlines can be computed. The sub-deadline of a task t_i is denoted by $d(t_i)$ and is calculated using Equation 9.

$$d(t_i) = (D \times max\, j \in \text{level}(t_i)\ EFT(t_j)) \setminus makespan \qquad (9)$$

In this equation, *makespan* specifies total timespan of the scheduling that has been completed in Algorithm 2.

After completing the first step, i.e., deadline distribution, Algorithm 1 checks the validity of the created schedule map. If it's not valid (i.e. its *makespan* exceeds the deadline), the second step must be done. The second step is finding the fastest schedule, and as it is mentioned before, this step is a modified version of the HEFT algorithm. Naive HEFT selects tasks in their priority order and assigns them to the earliest available resource. We modify this algorithm to consider sensitive tasks and different resource types.

The second step, which is the fastest schedule algorithm, selects tasks in their priority order, then based on the task's sensitivity chooses the resource type. There are two types of resources: private and public. The proposed algorithm selects private resources for sensitive tasks and public resources for normal tasks. If the task is insensitive, the resource is selected from the resources on the public cloud, and otherwise its resource is selected from the private cloud's resources. The pseudo-code is shown in Algorithm 3.

```
1  for all t_i ∈ G in nonincreasing order of rank value do
2  if t_i ∈ A then
3  Schedule t_i in u_j ∈ U such that EFT (t_i , u_j) is  minimum;
4   else
5   Schedule t_i in r_j ∈ R such that EFT(t_i , r_j) is  minimum;
6        end
7  end
8  Return (FastestScheduleMap);
```

Algorithm 3. Fastest scheduling algorithm

After the second step is completed, the validity of the scheduling is checked. If *makespan* is not less than or equal to user defined deadline, presenting a valid schedule is not possible. However, if schedule map is valid, the third step of the scheduling algorithm must be done.

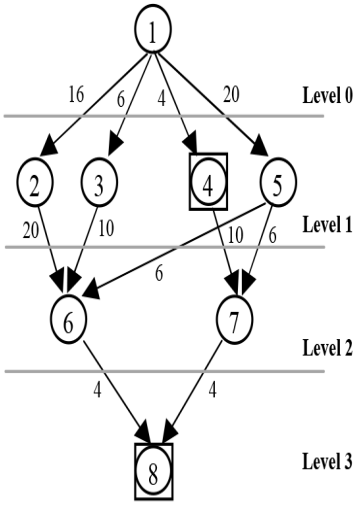
The third step of the algorithm is trying to reduce the cost of the schedule map. This schedule map is made by the second step. The third step also increases the *makespan* of this schedule and makes it closer to the deadline.

Algorithm 4 shows how the third step is done. In this step, called adjust schedule, whole resources of both private and public clouds are gathered into set *H*. Some of resources in *H* are assigned in the previous step.

Then, this algorithm reschedules normal tasks in set *A*. These tasks are scheduled on the public cloud, and the algorithm tries to move them to free resources on the private cloud, or cheaper resources on the public cloud. On each reschedule, the algorithm checks if all of task's successors are meeting their sub-deadlines, and if not so, the reschedule is not done.

```
Require: FastestScheduleMap
1  for all t_i ∈ A in nonincreasing order of rank value do
2      H ← R ∪ U;
3  while H ≠ ∅ do
4  EFT(t_i , h_j) ← ScheduleWithMinCost(H , t_i);
5   if EFT(t_i, h_j) <= d (t_i) then
6          Update EST , EFT for all successors of  t_i;
7   if CheckSuccForSubDeadlines()  then
8          ReAssign task t_i  to processor h_j ;
9            exit while;
10         end
11       end
12       H← H - {h_j};
13    end
14  end
```

Algorithm 4: Adjust schedule algorithm



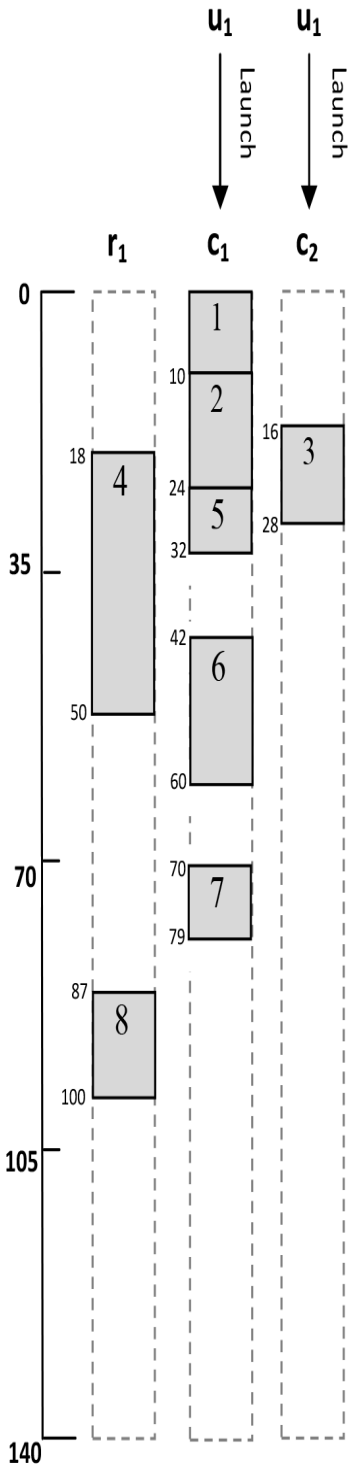
(a) An example of DAG with

Task	r_1	r_2	u_1	u_2	u_3
1	36	72	10	16	34
2	44	90	14	24	44
3	48	94	12	25	50
4	32	65	---	---	---
5	28	56	8	14	26
6	65	120	18	32	66
7	30	60	9	16	30
8	13	26	---	---	---

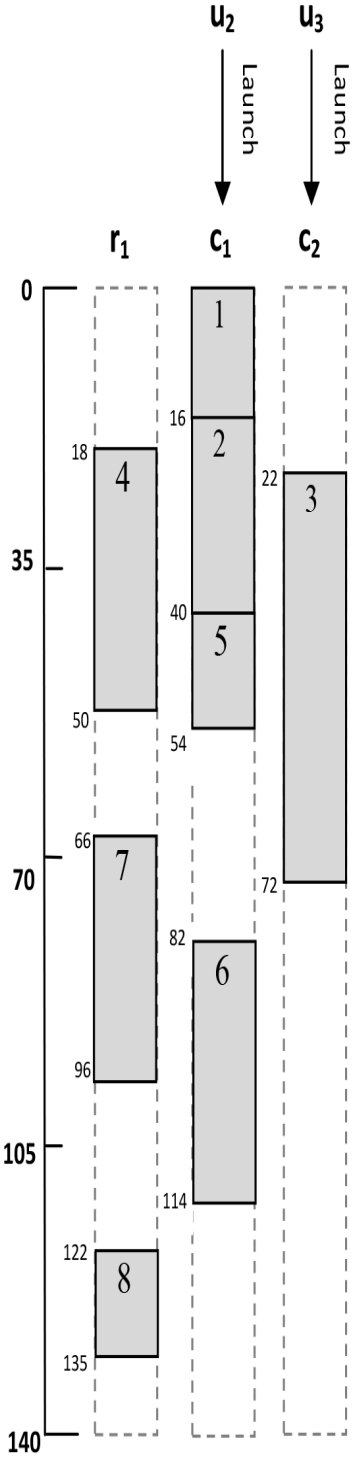
(b) Estimated execution times of tasks on different resources

Resource	r_1	r_2	u_1	u_2	u_3
r_1	0	2	0.5	0.5	0.5
r_2	2	0	0.5	0.5	0.5
u_1	0.5	0.5	0	1	1
u_2	0.5	0.5	1	0	1
u_3	0.5	0.5	1	1	0
Price	0	0	12	5	2

(c) Available bandwidth between different resources



(d) Schedule map obtained for DAG (a) By Fastest scheduling algorithm



(e) Schedule map of Adjust scheduling algorithm for *D*=140

Figure 2. Example of Scheduling On Hybrid Cloud Resources

This algorithm selects tasks in nonincreasing order of their rank. Then for each selected task t_i, Lines 3 to 13 are executed. In Line 4, function ScheduleWithMinCost() is called, which selects the cheapest available resource h_j on H and returns the earliest finish time of t_i on h_j. If there are more than one cheap resource, this function selects the one with the earliest finish time. If the returned *EFT* meets the task's sub-deadline $d(t_i)$ in Line 5, we update its successors in Line 6, and in Line 7, it is checked if all of task's successors meet their sub-deadlines. If every sub-deadline is met, task t_i is rescheduled in Line 8.

However, if this condition is not met, h_j is removed from H in Line 12, and this process is done again for the next cheapest resource.

Algorithm 4 is only performed for normal tasks, rescheduling them to cheaper resources with regards to their sub-deadlines. On the other hand, this algorithm does not reschedule sensitive tasks, and only can postpone their execution in order to hold task dependencies.

Figure 2. shows an example of scheduling in the hybrid system using WHPD. The DAG shown in Figure 2(a) is considered and the estimated execution time of each task on public resources (u_1, u_2 and u_2) and private resources (r_1 and r_2) are shown in Figure 2(b) Furthermore, Figure 3(c) gives the available bandwidths between different resources and the price of each timeslot (we assumed the timeslot be 60 time units). Assuming a deadline of 140, the outcome of schedule map using fastest schedule and adjusting schedule on the DAG in Fig. 2(a) is illustrated in Figure 2(d) and Figure 2(e), respectively.

5. Experimental Results and Evaluation

In order to evaluate the proposed algorithm, scientific workflows of Montage [19] and LIGO [20] are used. To validate the effectiveness of our proposed method, we compare our results with those of three state-of-the-art methods, namely, the HCOC, SPSS and the method proposed by Chopra and Singh [17]. All of these methods are scheduling methods on hybrid clouds; while the first one is a well-known scheduling method and all of them can potentially support sensitive tasks.

In order to adapt these methods to support sensitive tasks, for SPSS and the method proposed by Chopra and Sing, we do their algorithm for normal tasks and when it's time to schedule a sensitive task, a private resource with the earliest finish time is selected. For HCOC method, private tasks are not selected to be rescheduled, thus their first assignment is maintained.

Our experimental set includes 12 workflows, each of which must be scheduled on a hybrid cloud. The test architecture is consisting of 5 private resources that one of them has the computational capacity of 2, and others are 1. It is assumed that there are infinite resources on public cloud, with computational capacities of 1, 2 and 4, while their prices are 1, 2 and 4 respectively.

In order to assign a deadline for each workflow, firstly we assign every task of the given workflow on a resource with computational capacity of 1. Then considering all communications, we calculate the *makespan* of the workflow. Then we multiply this *makespan* by a variable coefficient called *deadline factor*, to get different deadlines for the workflow. In our experiments we change the *deadline factor* from 1 to 4, with step 1.

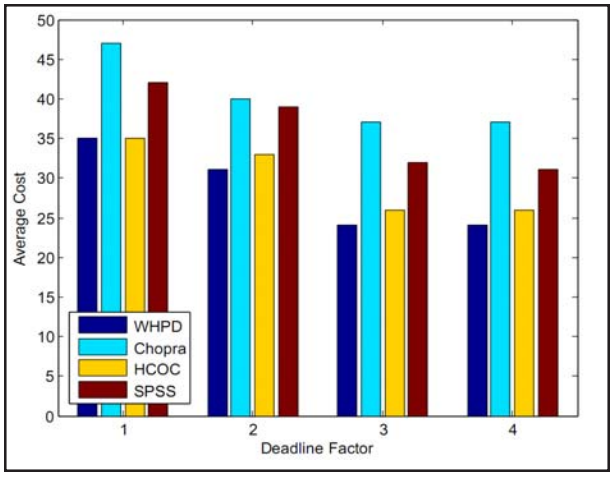


Figure 3. The relationship between Cost and Deadline Factor

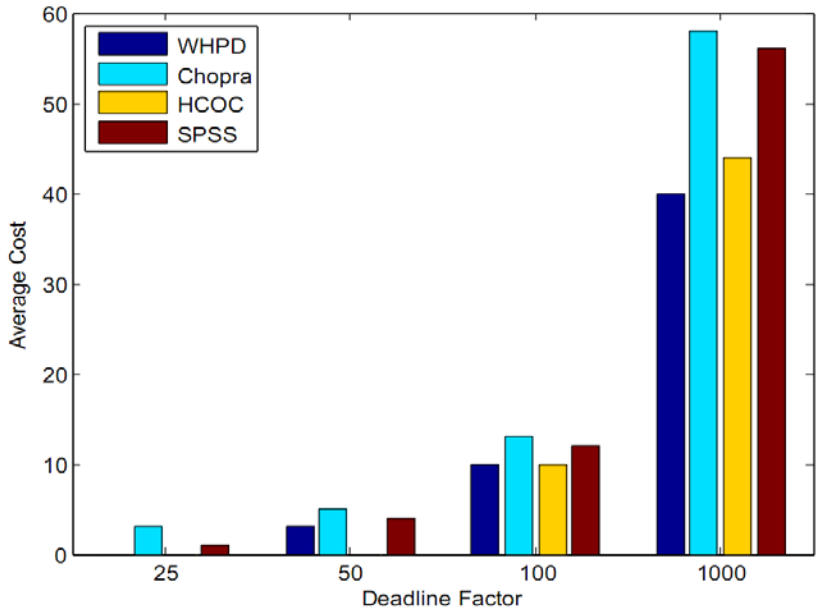


Figure 4. The relationship between Cost and Workflow size

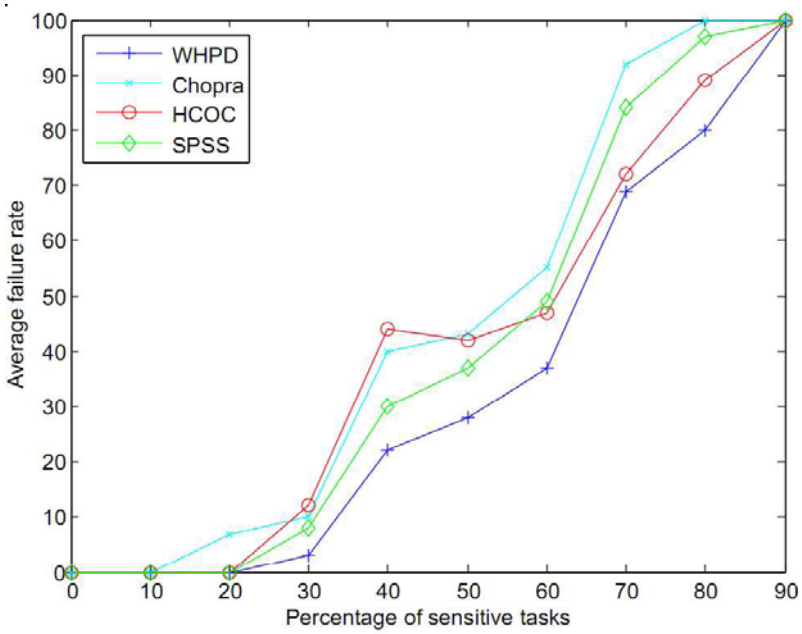


Figure 5. The relationship between Fail and Percentage of Sensitive tasks

The relation between deadline and cost in different methods are compared in Figure 3. using Montage workflow and different *deadline factors*. In this comparison none of the workflows consists of sensitive tasks, and each method tries to reduce the cost of running workflow under the given deadline. Another comparison is made using a similar setup, with the only exception being the inclusion of sensitive tasks. To this aim, we have changed the Chopra method to support sensitive tasks. In this experiment, different sizes of Montage workflow are examined. This experiment is demonstrated in Figure 4. The reason for the significant difference between cost of the workflow with size 1000 and smaller workflows is the limitation of resources on private cloud. This causes the scheduler to use costly resources from public cloud.

The proposed method schedules sensitive tasks only on private cloud, and this limitation may lead to schedule's failure. In order to investigate this, we change the percentage of sensitive tasks and check if the scheduler fails. Results are depicted in Fig. 5.

6. Conclusion

In this paper a method considering quality of service was proposed for scheduling workflows with the minimum cost and before the deadline imposed by the user. The presented algorithm guarantees the execution of sensitive tasks on the private clouds while the scheduled workflow is completed before the user's deadline (for cases that are possible). Based on the conducted experiments, the proposed algorithm could achieve at least 5% reduction in the cost of using public clouds with at least 7% improvement in successful scheduling compared to HCOC, SPSS and the method proposed by Chopra and Singh.

References

[1] Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., Brandic, I. (2009). Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility, *Futur. Gener. Comput. Syst.*, 25 (6) 599–616.

[2] Subashini. S., Kavitha, V. (2011). A survey on security issues in service delivery models of cloud computing, *J. Netw. Comput. Appl.*, 34 (1) 1–11.

[3] Andrew, A. M. (2012). Cloud computing: views on Cybersyn, *Kybernetes*, 41 (9) 1396–1399.

[4] Zissis, D., Lekkas, D. (2012). Addressing cloud computing security issues, *Futur. Gener. Comput. Syst.*, 28 (3) 583–592.

[5] Gens, F. (2009). New idc it cloud services survey: top benefits and challenges, *IDC Exch.*, 2009.

[6] Zhou, Z., Zhang, H., Du, X., Li, P., Yu, X. (2013). Prometheus: Privacy-aware data retrieval on hybrid cloud, *In*: *INFOCOM*, 2013 Proceedings IEEE, 2643–2651.

[7] Jamil, D., Zaki, H. (2011). Cloud computing security, *Int. J. Eng. Sci. Technol.*, 3 (4) 3478–3483.

[8] Shetty, S. (2013). Gartner Says Cloud Computing Will Become the Bulk of New IT Spend by 2016.

[9] Michael, R. G., David, S. J. (1979). Computers and intractability: a guide to the theory of NP-completeness, *WH Free. Co., San Fr.*, 1979.

[10] Wieczorek, M., Hoheisel, A., Prodan, R. (2009). Towards a general model of the multi-criteria workflow scheduling on the grid, *Futur. Gener. Comput. Syst.*, 25, 237–256.

[11] Rahman, M., Li, X., Palit, H. (2011). Hybrid heuristic for scheduling data analytics workflow applications in hybrid cloud environment, *In*: Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on, 966–974.

[12] Rahman, M., Venugopal, S., Buyya, R. (2007). A dynamic critical path algorithm for scheduling scientific workflow applications on global grids, *In*: *e-Science and Grid Computing, IEEE International Conference on*, 35–42.

[13] Bittencourt, L. F., Madeira, E. R. M. (2011). HCOC: A cost optimization algorithm for workflow scheduling in hybrid clouds, *J. Internet Serv. Appl.*, 2, 207–227.

[14] Bittencourt, L. F., Madeira, E. R. M. (2008). A performance-oriented adaptive scheduler for dependent tasks on grids, in *Concurrency Computation Practice and Experience*, 20, 1029–1049.

[15] Topcuoglu, H., Hariri, S., Wu, M. (2002). Performance-effective and low-complexity task scheduling for heterogeneous computing, *Parallel Distrib. Syst. IEEE Trans.*, 13 (3) 260–274.

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

[17] Chopra, N., Singh, S. (2013). Deadline and cost based workflow scheduling in hybrid cloud, in *Advances in Computing, Communications and Informatics (ICACCI)*, 2013 *International Conference on*, 2013, 840–846.

[18] Annapureddy, K. (2010). Security challenges in hybrid cloud infrastructures, *Aalto Univ.*

[19] Gil, Y., Deelman, E., Ellisman, M., Fahringer, T., Fox, G., Gannon, D., Goble, C., Livny, M., Moreau, L., Myers, J. (2007). Examining the challenges of scientific workflows, *IEEE Comput.*, 40, (12) 26–34.

[20] Deelman, E., Kesselman, C., Mehta, G., Meshkat, L., Pearlman, L., Blackburn, K., Ehrens, P., Lazzarini, A., Williams, R., Koranda, S., GriPhyN and LIGO, building a virtual data grid for gravitational wave scientists, *In*: High Performance Distributed Computing, 2002. HPDC-11 2002. Proceedings. 11th IEEE International Symposium on 2002, 225–234.