Privacy-Aware Scheduling SaaS in High Performance Computing Environments

Shaghahyegh Sharif, Paul Watson, Javid Taheri, Surya Nepal, and Albert Y. Zomaya, *Fellow, IEEE*

Abstract—Hybrid clouds have gained popularity in recent times in a variety of organizations due to their ability to provide additional capacity in a public cloud, to augment private cloud capacity, when it is needed. However, scheduling distributed applications' jobs (e.g, workflow tasks) on hybrid cloud resources introduces new challenges. One key problem is the danger of exposing private data and jobs in a third-party public cloud infrastructure, for example in healthcare applications. In this article, we tackle the problem of designing workflow scheduling algorithms to meet customers' deadlines, while not compromising data and task privacy requirements. Our work is different from most studies on workflow scheduling where the main goal is to achieve a balance between desirable, yet incompatible constraints, such as meeting the deadline and/or minimizing the execution time. Although many others have addressed the trade-off between cost and time, or privacy and cost, their work still suffers from an insufficient consideration of the trade-off between privacy and time. To address such shortcomings in the literature, we present a new SaaS scheduling broker composed of MPHC-P1, MPHC-P2, and MPHC-P3 policies to preserve privacy while scheduling the workflows' tasks under customers' deadlines. We evaluated our approach using real workflows running on a VMware based hybrid cloud. Results demonstrate that under our scheduling policies, MPHC-P2 and MPHC-P3 are promising in time-critical scenarios by reducing the total cost by 10-20% compared to alternatives. Overall, results show that our approach is efficient in reducing the cost of executing workflows while satisfying both their privacy and deadline constraints.

Index Terms—Hybrid Cloud, SaaS, Workflow Scheduling, Privacy Preserving

+

1 Introduction

Many organizations rely on scalable computing infrastructures that allow them to adapt to changes in business and customer demands. Cloud technology has become accepted as a cost effective method for delivering services [1], due to the potential it offers for organizations to be agile in launching new services and meeting changing requirements. Workflows are commonly used to implement business processes on clouds and elsewhere [2]. They are also extensively applied in a diverse range of other areas including astronomy, bioinformatics, and physics.

Assigning workflow tasks to available computing resources in order to satisfy customer demands is known as workflow scheduling, and is a classical optimization problem. This NP-hard problem has a profound history and has been the target of extensive studies, including in the context of cloud computing [3], [4], [5], [6], [7], [2], [8], [9], [10]. Scheduling workflows in the cloud is subject to various constraints, and fulfilling one might result in failing to meet others. This can happen for several reasons, such as their conflicting behavior, the complex nature of workflows, and the pay-as-you-go pricing model of most current commercial cloud providers [2]. Hence, many studies have been done on workflow scheduling to achieve a balance between these desirable but incompatible objectives/constraints. Cloud performance benchmarks are based on Service Level Agreements (SLAs) between resource providers and users in most enterprises; they include, but are not limited to, diverse quality of service parameters such as deadline, budget, reliability, availability, and privacy. In this study, our objective is to minimize the cost of workflow scheduling while satisfying two constraints: preserving privacy and meeting a compulsory deadline.

We define a multi-level privacy constraint with a high degree of granularity for both task and data without a necessity to have a priori knowledge of workflows data structure. The novelty of our work is solving the combinatorial optimization problem of workflow scheduling, while satisfying the constraint of time by adding another layer of complexity to guarantee the preserving of a multi-level privacy. Our proposed scheduler in this work considers privacy level of tasks and data, according to individual user requirements, to schedule workflow tasks among available computing resources without loosing the data utility. Because of the extra added layer, our scheduling model is much more complex when compared with similar approaches in existing studies [11]. There already exist several single-level privacy-aware schedulers to sanitize data, by using sensitivity tags [12],[13], where constraints only specify whether

- *Shaghayegh Sharif is with the Center for Distributed and High Performance Computing, School of Information Technologies, The University of Sydney, NSW 2006, Australia, and also associated with ICT cloud research group, CSIRO, NSW, Australia. Email: ssn@it.usdy.edu.au*
- *A.Y.Zomaya is with the University of Sydney, Australia. Email: albert.zomaya@sydney.edu.au.*
- *J. Taheri is with Karlstad University, Sweden. Email: javid.taheri@kau.se*
- *S. Nepal is with CSIRO, Australia. Email: Surya.Nepal@csiro.au*
- *Paul Watson is with Newcastle University, England. Email: paul.watson@newcastle.ac.uk*

tasks/data are allowed to be scheduled in public clouds or not. Our primarily motivation is that while this is clearly important for many classes of applications – e.g., those in healthcare, where patient confidentiality might be compromised– the simultaneous fulfillment of these two constraints has not been addressed in previous works. To the best of our knowledge, our approach in this article is the first to simultaneously consider all aforementioned privacy contains with granule definition of privacy levels for both workflows tasks and cloud resources.

In nutshell our contribution is a time-guarantied scheduling approach to concurrently minimize the cost of scheduling and satisfying the privacy requirement of the workflow. We believe satisfying both privacy and deadline constraints are crucial because cloud privacy has becoming a major area of concern when scheduling over federations of private, community (the cloud infrastructure for restrictive use by a specific community of users from organizations that have shared concerns –these have become known as Hybrid Clouds (HCs) [14]. Elasticity in response to increasing demand is a key feature of HCs. It gives private cloud users the ability to migrate the execution of part of their workflows from the private to public clouds when demand increases, and they do not have sufficient private cloud resources to meet the deadlines of all requests. However, this important feature of a HC can also introduce new problems as applications cannot rely on the traditional assumptions about data center security [13], [11]. There are often privacy concerns with the potential deployment of data and tasks on a shared cloud infrastructure owned and managed by an external, third-party provider, possibly in a different legal jurisdiction. These issues can be a real barrier to organization wishing to exploit the benefits of public clouds federated with their own private cloud in a HC. In this study, we aim to address this problem by introducing a new approach to establishing and meeting SLAs covering both the preserving of privacy and meeting deadlines.

Existing studies on workflow scheduling commonly address up to two conflicting criteria (e.g., time and cost) to provide maximum customer satisfaction. They either attempt to minimize the workflow execution time or focus on the minimization of cost while trying to meet the application deadline [6], [15], [10], [2].

To address the limitations of previous research, we propose a new Software as a Service application (SaaS) to schedule workflows while satisfying their component tasks required deadline and multi-level privacy constraints in HCs. Our design differentiates between public and sensitive data and tasks by assigning private tasks to private clouds and delegating non-private tasks to public commercial clouds using deadline and privacy aware scheduling policies. Our contribution to this field is to design, implement, and evaluate a multi-criteria workflow scheduler to minimize workflow execution costs while satisfying all users' privacy and deadline requirements; our contributions can be highlighted as:

1) Defining the problem of resource provisioning for workflows under privacy and deadline constraints, while optimizing cost.
2) Applying two privacy preserving policies (Multi-terminal Cut, BLP) and studying the difference.
3) Applying three deadline awareness policies (MPHC-P1, MPHC-P2, and MPHC-P3) and studying the effectiveness of their result under various billing cycles in HCs.
4) Performing all evaluations using well-known scientific workflows (Montage, LIGO, Epigenomics, and Cybershake), plus two sample healthcare workflows.

As a continuation of our previous work [16], here we present more detailed performance evaluations, including, but not limited to deep discussion and analysis of our new results in this work. We also report on new results and discussion on different privacy preserving policies and resource billing cycles, as the important aspects of scheduling and resource provisioning in HCs. The remainder of the paper is organized as follows. Section 2 reviews the related work. We elaborate the problem and our case studies in section 3. Section 4 details our proposed solution. We describe our approach in section 5. Evaluation and results are presented in section 6. Discussion of our results and the conclusions drawn from our work are presented in sections 7 and 8, respectively.

2 Related Work

Resource provisioning in large-scale heterogeneous environments such as clouds has been widely studied under various quality of service (QoS) parameters [17], [9], [18], [15]. These are commonly categorized into budget, deadline, reliability, availability, execution time, and security. In the current literature, it is more common that the cost is specified as an objective to be minimized while satisfying a fixed deadline. There exist several studies that have considered workflow privacy in the use of cloud technology [19], [20], [21]. However, they have failed to consider other constraints such as user budget and deadline in deploying workflows in cloud environments.

Although the problem that we are trying to solve is categorized in a combinatorial optimization, there are alternative approaches to over come this problem using common privacy approaches [22], [23], [24], [25] such as differential privacy [26], [27], [28], [29], data encryption [30], or anonymization [19] in cloud environments to preserve the privacy. There are limitations in using these common privacy approaches. Considering differential privacy approach, utility is a challenge as statistical properties change as adding more noise to the data. Also, Many non-trivial differential privacy algorithms require really large datasets to be practically useful [31]. Processing encrypted data in HCs is challenging because most existing applications and workflow managers run mostly on unencrypted data [21]. Moreover, when already anonymized data are combined with new data, a violation of privacy requirements or over-anonymization might take place. Existing methods to address this problem have attempted to re-anonymize all of the updated data [21] which may lead to inefficiency and vulnerability to privacy attacks [32], [21] especially in HCs. Alongside these studies, there are other studies that focus on data computation without encryption (homomorphic encryption); these techniques are however rather expensive and impractical with regard to efficiency [21].

Regarding workflow scheduling in a cloud environment, there are two types of assumptions considering the privacy: 1) a workflow that is entirely private, and 2) a workflow that is partially private (a few tasks or data items are private). The first assumption is outside the scope of this work because the whole workflow is private, and thus cannot be executed anywhere except on private clouds. The second type, however, is more complicated due to mixed privacy requirements for tasks and/or data where tasks/data can be scheduled/replicated on either private or public cloud resource; our work is directly addressing this issue. To this end, Smanchat et al. [17] categorized the privacy constraint into two subgroups: single-level and multi-level. The single-level constraint simply specifies whether a dataset (and tasks that are using that particular dataset) requires privacy. This is also assumed by SABA [13], [12] which differentiates between "movable datasets (without privacy)" and "immovable datasets (with privacy)". Most of the other workflow scheduling techniques usually assume the data of the first type is transferable as required. The data of the second type are not allowed to be transferred and duplicated. On the contrary, the multi-level privacy model assumes that privacy requirements can be specified at many levels [11], [16]. As opposed to the previous subgroup (single level), workflow scheduling is more complicated as datasets and tasks require specific privacy levels to be executed on public instances whose images are installed with trusted software or services [17].

Apart from the privacy concern, there are also many studies on workflow scheduling under deadline and budget constraints that have neglected privacy in HCs. We classified these studies into two groups. The first group mostly considered minimizing the execution time under budget constraints [3], [4], [5], while the second group considered both budget and deadline at the same time [6], [7], [2], [8], [9], [10]. Because workflows are modeled as a Directed Acyclic Graph (DAG), the majority of these algorithms have taken advantage of the critical path for scheduling tasks. One of the primary works in workflow scheduling is the famous Heterogeneous Earliest Finish Time (HEFT) [33]. This algorithm that was originally designed for heterogeneous computing systems is inherently unaware of special characteristics of cloud computing and utility grid paradigms. HCOC [3] is a cost-optimized scheduling algorithm under deadline constraints. This algorithm is able to minimize execution costs in public clouds by extending the desired deadlines. Abrishami [2] presented IC-PCP and IC-PCPD2, two algorithms for scheduling workflows in an IaaS cloud environment under deadline constraints. For ensemble workflow scheduling, three algorithms: DPDS, WA-DPDS and SPS were proposed by [34] for resource provisioning in IaaS clouds. They maximized the number of user-prioritized workflows that can be completed within the desired time and budget. The MDP algorithm in [8] partitioned a DAG into sections to schedule tasks within their deadlines, while decreasing the overall cost. Additionally, authors of both [35] and [36] attempted to minimize data movement and optimize the cost for workflow scheduling.

The challenge presented in this article is different from all aforementioned algorithms regarding their shortcomings in constraint integration, and also impractical and inefficient data and task privacy methods in HCs. We contribute to this field by not only considering the deadline and budget, but also multi-level privacy of data and tasks in workflows.

3 Problem Description

To elaborate more on the motivations behind our study, we describe the system architecture of our SaaS scheduler broker, along with sample workflows. We assume that our HC provides a SaaS application with an underlying infrastructure similar to the Amazon Elastic Compute Cloud (EC2) [37] or Microsoft Azure [38], where the computing resources can be set up to execute workflows. Davidson et al. [20] suggested following privacy concerns when considering workflows: first, workflow's owners' demand to keep some intermediate data private (e.g., social security numbers, a medical records, or financial information). Second, the functionality of some workflow tasks is preferred to be private in order to protect the methods of those tasks. Third, the entire workflow structure may also be proprietary, for example, exposing how data is passed between tasks may reveal too much of the workflow structure. In this study, we consider the first and second privacy models, which include task and data privacy.

3.1 System Architecture

Our architecture utilizes scheduler broker software (Fig. 1) to manage instance provisioning for input workflows. Users submit their workflow applications through the *Workflow Registry*. The submitted application profile includes task specifications and their dependencies. The *Scheduler Module* makes decisions about provisioning of HC's instances as well as allocation of resources to the submitted workflows regarding customers' SLAs.

Since workflows are deadline-guaranteed, the scheduler might outsource part of a workflow tasks onto public/community clouds. In other words, if the private

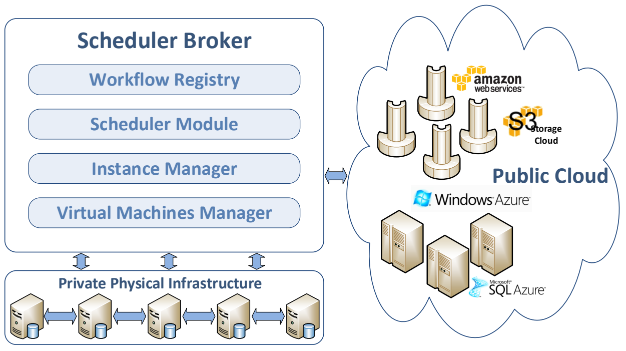


Fig. 1. Scheduler broker (SaaS in a hybrid cloud)

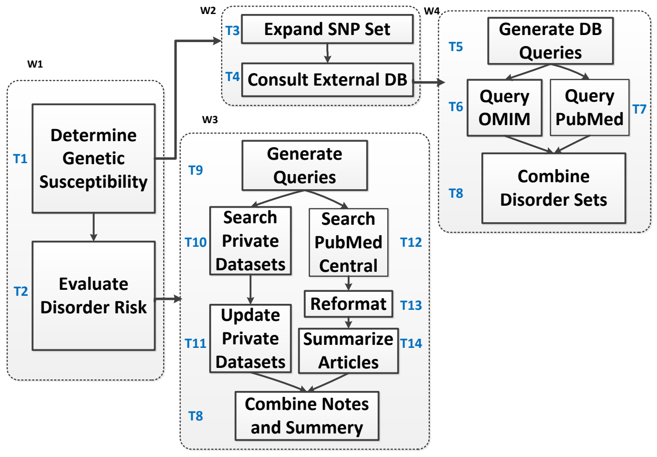


Fig. 2. Disease susceptibility workflow specification [20]

cloud cannot meet a workflow's SLA, the scheduler partitions workflow tasks into sub-graphs and distributes them among both private, community and public instances. In this case, the cost and delay of transferring data between separated sub-graphs of the workflow is considered. The scheduler addresses the trade-offs in selecting appropriate instances to satisfy SLA constraints of time, cost, and privacy. The objective of the scheduler is to maximize the profit of the private business enterprise without violating any workflow's SLA. The output of the scheduler algorithm is sent to the *Instance Manager* component to instantiate/terminate on hired/owned instances. This component keeps track of all available instances in the private and public clouds.

3.2 Workflow Samples

In this section, we present two sample workflows to highlight the key existing challenges in this area of research.

3.2.1 *Disease Susceptibility Workflow*

Assume an enterprise (such as a bioinformatics company) owns a sophisticated proprietary workflow manager which collects and analyzes data related to a susceptibility to a disease. The enterprise has access to three type of resources in its HC; 1) private instances assigned to the each department, 2) common instances available to all departments inside the enterprises, and 3) public instances from commercial cloud providers. Since the community cloud is provisioned for exclusive use by a specific organizations' users that have shared concerns (e.g., mission, security requirements, and policy), we assumed the second type of the resources is the same as the community cloud. Considering Fig. 2, we show a simplified workflow with privacy and deadline constraints to be executed in the company's HC. Tasks such as $T1$, $T2$, and $T10$ have to be executed on private instances. $T3$, $T4$, $T13$, and $T14$ can be run on any departments instances inside the organization. Finally, $T5$, $T6$, $T7$, $T8$ are eligible to be run on any available instances.

Regarding the SLA (deadline and privacy) between customers and bioinformatics company, workflow tasks have to be deployed on private instances, shared instances within the enterprise, or public providers regarding to tasks/data privacy requirement. The challenge that is addressed here is "How does the company deploy a workflow on a HC when there are not enough private resources to execute the workflow before its deadline?". Our solution to preserve the privacy while deploying these workflows on a HC is to utilize a broker SaaS to manage instance provisioning for executing workflows similar to the one presented in Fig. 1.

3.2.2 *Medical Data Analysis Workflow*

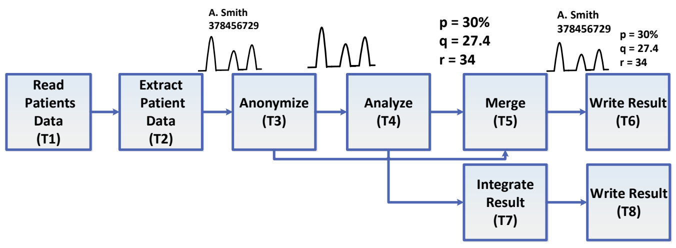


Fig. 3. An example medical data analysis workflow

The second sample is a real medical research application conducted in Newcastle University (UK) that analyzes data from patients' activity sensors [11]. As shown in Fig. 3, this workflow analyzes data for each individual patient, and has an assigned deadline. Initially, $T1$ reads patient data over a specific time period. The input data for $T2$ is a file with a header to individually identify each patient, and $T3$ is an anonymizing service to exclude the header. $T1$, $T2$, and $T3$ are private and should only be executed on local resources (e.g., health-care research department) that are allowed to access private data. The summary of the results without any private information is aggregated -with other patients' results- and recorded by $T7$ and $T8$, respectively. $T1$, $T2$, $T5$, and $T6$ are private and must be executed on local private resources. $T3$ and $T4$ are less sensitive tasks and can be executed anywhere within university departments instances (similar to the community cloud). Analyzing the activity data is computationally expensive and it would benefit from the cheap and scalable resources that are available on public and community clouds when the private cloud does not have sufficient resources to finish the workflow before its deadlines. As a solution to overcome the privacy barriers on deploying the workflow's executions on public clouds, the service provider utilizes the broker shown in Fig. 1 to execute the workflow using a HC.

4 System Model

In this section, to understand how the SaaS scheduler broker works, we provide a comprehensive description of our privacy model, the broker's input jobs, and the underlying infrastructure of resources in HCs. In our scenario in order to create a hybrid cloud, three distinct cloud infrastructures (private, community, and public) are integrated so that they are still unique entities, but are bound together by a standardized broker technology.

4.1 Privacy Model

In this study, the privacy constraint is tied to components (data and tasks) of a workflow. Three levels of privacy are considered for workflow tasks and data [20] as $\Gamma_t = \{\tau_{n1}, \tau_{n2}, \tau_{n3}\}$: where (1) τ_{n1} represents tasks that can be deployed on public, community, and private resources without any restriction (inside or outside of the organization), (2) τ_{n2} represents tasks that can be deployed on private instances and community clouds (only inside one department and some selected resources of other departments in one organization), and (3) τ_{n3} represents tasks that can be deployed only on private instances (only inside of a department). We also assign privacy tags to resources noted as $\Gamma_s = \{\tau_{s1}, \tau_{s2}, \tau_{s3}\}$. τ_{s1}, τ_{s2}, and τ_{s3} are public, community, and private instances, respectively. Here, relation R_s maps each resource to the relevant privacy privilege $R_s : HC_S \rightarrow \Gamma_s$. A resource which is tagged with (1) τ_{s1}: the public cloud instances (eligible to host tasks with privacy τ_{n1}), (2) τ_{s2}: the community cloud instances (eligible to host tasks with privacy τ_{n1} and τ_{n2}), and (3) τ_{s3} the private cloud instances (eligible to host all tasks without any restriction). Table 1 demonstrates the allocation map based on privacy privileges. Symbol '✓' in each cell represents an eligible allocation; '×' represents the opposite.

TABLE 1
Privacy Map Allocation

	τ_{n1}	τ_{n2}	τ_{n3}
τ_{s1}	✓	×	×
τ_{s2}	✓	✓	×
τ_{s3}	✓	✓	✓

4.2 Resource Model

Our resource model is assumed to be similar to Amazon's Elastic Compute Cloud (EC2) [37] where virtual machine (VM) instances are leased on demand and are billed based on the time intervals they are leased to run workflows. Private, community, and public service providers offer different computation services $S_{prv} = \{s^1_{prv}, \ldots, s^m_{prv}\}$, $S_{com} = \{s^1_{com}, \ldots, s^n_{com}\}$, and $S_{pub} = \{s^1_{pub}, \ldots, s^n_{pub}\}$; where our HC resource pool is $HC_S = \{S_{prv} \cup S_{com} \cup S_{pub}\}$. Each service has its individual specifications: CPU type, price, and the privacy access level. In this study, we assume that resources on private clouds cost less than those with identical specification in public clouds; this assumption is made because workflow managers encounter extra costs when outsourcing customer's workflows to public resources.

4.3 Application Model

Our target applications are DAG type workflows, where nodes represent computational tasks, and edges represent data and task dependencies. Each workflow application is defined as: $J = \{G, SLA\}$ where $G = (N, E)$ is the DAG. $N = \{n_1, n_2, n_3, ..., n_n\}$ is the set of tasks with n_i denoting the i^{th} task of the graph, and $E_i = \{(n_j, n_k) \mid n_j, n_k \in N\}$ is the set of links between tasks. $SLA = (D, P)$ is the SLA contract where D is the workflow's deadline and P is the set of privacy tags for tasks in N. In a given DAG, a task without any parent is called a n_{entry} *task*, and a task without any child is called a n_{exit} *task*. We add a n_{entry} and a n_{exit} to the beginning and the end of each workflow to make sure each workflow has only one input and one output node. These tasks have zero execution time and they are connected with zero-weight dependencies to the actual entry and exit tasks. Each task in a DAG has three scheduling attributes: (1) **Earliest Start Time (**EST**)**, (2) **Earliest Finish Time (**EFT**)**, and (3) **Latest Finish Time (**LFT**)**. EST for each task n_i is computed before scheduling a workflow; EST (Eq. 1) and EFT (Eq. 2) are calculated as follows:

$$\begin{aligned} &EST(n_{entry}) = 0 \\ &EST(n_i) = \max_{n_p \in n_i's\ parents} \{EST(n_p) + MET(n_p) + TT(e_{pi})\} \end{aligned} \tag{1}$$

where n_p is one of n_i's parents and $TT(e_{pi})$ is the data transfer time between n_p and n_i. $MET(n_i)$ (Minimum Execution Time of a task n_i), is the execution time of task n_i on a service $s_j \in S$ which has the minimum execution on sj between all available resources in HC_S.

$$EFT(n_i) = EST(n_i) + MET(n_i) \tag{2}$$

The EFT of task n_i is the sum of its EST and minimum execution time. The Latest Finish Time, LFT, is calculated as follows (Eq. 3):

$$LFT(n_{exit}) = D$$
$$LFT(n_i) = \min_{n_c \in n_i' s\ children} \{LFT(n_c) - MET(n_c) + TT(e_{ic})\} \quad (3)$$

where n_c is one of n_i's children and $TT(e_{ic})$ is the data transfer time between n_i and n_c. The LFT of an unscheduled task n_i is defined as the latest time at which n_i can finish its computation so that the whole workflow can finish before the user's defined deadline (D).

Fig. 4 presents values of EST and EFT for each node in a sample workflow from Fig. 3, running on the fastest resource with the requested deadline as 300.

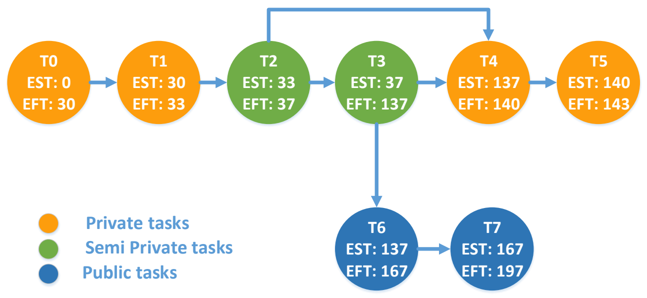


Fig. 4. Healthcare workflow with deadline 300

4.4 Cost Model

The cost model used in this study is derived from our previous work [16]. It calculates the trade-off between expenditure and revenue, considering both private, community, and public instances, to estimate the profit of a SaaS provider (Eq. 4). Here, the community cloud's instances are treated the same as private instances in terms of the cost model. The public cost is the price of leasing resources from a public provider including bandwidth and storage (Eq. 5). We ignored overheads incurred by workflow management systems as they are strongly dependent on the particular technology for workflow management in use, and fairly negligible as compared with the overall execution time of multiple workflows in such systems. The revenue (Eq. 6) from private or public resources depends on the number of private and public instances acquired over their time intervals multiplied by their corresponding prices, in addition to the cost of network bandwidth in a private and a public cloud.

$$profit(\Delta t) = rev_{total}(\Delta t) - cost_{total}(\Delta t) \quad (4)$$

$$cost_{total}(\Delta t) = cost_{prv}(\Delta t) + cost_{pub}(\Delta t) \quad (5)$$

$$rev_{total}(\Delta t) = rev_{prv}(\Delta t) + rev_{pub}(\Delta t) \quad (6)$$

5 MPHC, The Scheduler Algorithm

In this study, our approach is the extension of our previous research [16]. Previously, we introduced the "Multiterminal-Cut for Privacy in Hybrid Clouds (MPHC)" algorithm in order to schedule workflows on HCs built with a single scheduling policy. In this article, our method is composed of two main modules: 1) *privacy preserving module* to satisfy required privacy levels for all tasks/data, and 2) *scheduling module* to acquire eligible resources for executing tasks before their deadlines. We developed two policies in the privacy preserving module (Multiterminal-Cut and BLP), and three policies in the scheduling module (MPHC-P1, MPHC-P2, and MPHC-P3). The output of the privacy preserving module is a list of sub-DAGs, which is the input for the scheduling module. Alg. 1 represents the main procedures of our proposed algorithm in this work. All modules, including their policies, are detailed in the following subsections.

Algorithm 1 Main Scheduler

```
1: procedure MAINSCHEDULER($G(N, E), D$)
2:     $subDAGs \leftarrow$ PrivacyModule(PrivacyPolicy)
3:     Initialize available computation resources
4:     compute $EST(n_i)$, $EFT(n_i)$ and $LFT(n_i)$ for all $n_i$
5:     SchedulingModule($subDAGs$, SchedulingPolicy)
6: end procedure
```

Step 2 of Alg. 1 invokes the privacy preserving module to enforce privacy policies, and produces the input for the scheduling module. The generated sub-DAGs are then used in Step 5 to produce the final allocation of tasks as complied with scheduling policies for HC instances.

5.1 Privacy Preserving Module

Here, two alternative policies are used to preserve privacy of tasks/data in workflows: 1) Multiterminal-Cut, and 2) Bell-LaPadula (BLP). In both policies, a workflow is partitioned into a number of sub-DAGs such that in each sub-DAG, tasks' privacy tags are labeled in range of one of these groups listed as; (τ_{n1} and τ_{n2}), (τ_{n1} and τ_{n3}) or (τ_{n1}). In our previous studies, we limited the privacy model to three levels and used only the Multiterminal-Cut approach to preserve privacy. Because this privacy model does not always cover all workflows' privacy requirements –such as when the privacy of the input data for a task is higher (τ_{n3}) than a task itself (τ_{n1} or τ_{n2})–, we made a realistic assumption that input data of a task can not have a higher privacy requirement than the task itself; i.e. it must have the same or lower privacy level than its relevant tasks. In this study, we use Bell-LaPadula (BLP) as a second alternative to preserve privacy and enhances the privacy model of our previous works. The BLP is a multi-level access control model which is extended and adopted to preserve privacy in workflows [11]. Multiterminal-Cut results in only one valid deployment option (only one list of sub-DAGs), whereas BLP delivers several deployment options, especially for large-sized workflows (several lists of sub-DAGs). For this reason, although it is desirable to apply a heuristic in BLP to limit the number of valid deployments [39], we did not target it in this work because it is beyond the scope of our work in this paper. In the next two subsections, each privacy method is clarified in detail.

5.1.1 Multiterminal-Cut Method

The Multiterminal-Cut algorithm applies an isolating cut heuristic for a workflow considering some of its nodes as terminals: for each terminal node n_i, it computes a minimum cost cut to separate it from the remaining terminals and produces the union of the $k-1$ lightest cuts as output [35]. As a result of applying the Multiterminal-Cut, a workflow is partitioned into k numbers such that each sub-DAG contains one of the terminal nodes ($k = 3$ in this study). Nodes with similar privacy levels are first contracted (Node-Contraction Alg. 2), and then considered as a terminal. We assume all tasks that require specific private data are scheduled on the same resources.

Algorithm 2 Node-Contraction Algorithm [35]

```
 1: procedure NODECONTRACTION($G(N, E)$, $n_i$ and $n_j$)
 2:     Create new node $n_q$
 3:     Let $n_q^{runtime} = n_i^{runtime} + n_j^{runtime}$
 4:     if there is an edge between ($e_{iu} \leftarrow (n_i, n_u)$) OR ($e_{ju} \leftarrow (n_j, n_u)$) then
 5:         create a new edge ($e_{qu} \leftarrow (n_q, n_u)$)
 6:         Let $weight(e_{qu}) = weight(e_{iu}) + weight(e_{ju})$
 7:     end if
 8:     if there is an edge between ($e_{ui} \leftarrow (n_u, n_i)$) OR ($e_{uj} \leftarrow (n_u, n_j)$) then
 9:         create a new edge ($e_{uq} \leftarrow (n_u, n_q)$)
10:         Let $weight(e_{uq}) = weight(e_{ui}) + weight(e_{uj})$
11:     end if
12: end procedure
```

Node-Contraction algorithm (Alg. 2) is composed of two steps; it recursively invokes the basic node contraction algorithm. During the first step, every node with privacy tag of τ_{n1} (private) or τ_{n2} (semi-private), and only one node with τ_{n3} (public) in the input DAG is contracted with all nodes that are directly connected to produce a single "super node". The second step is to contract all super nodes with the same privacy level into a larger super node, similar to the first step. Through such basic node contraction methods, two connected nodes are replaced by a new node; the weight of the new super node is calculated as the total weight of all its contracted nodes. Edges connecting to the contracted nodes are also replaced by new edges; their costs are assigned accordingly. After contracting all mentioned nodes, there will be only one super node for each privacy level (τ_{n1}, τ_{n2}, and τ_{n3}) in a DAG. Upon applying the Multiterminal-Cut, the new constructed DAG is partitioned into $k = 3$ sub-DAGs. Each sub-DAG contains a super node, while the total weight of all cut edges is minimized. Finally, for each sub-DAG all the contracted nodes are extracted to their original format and as a result each sub-DAG contains nodes with similar privacy tags. We should keep in mind that nodes with a public privacy tag (τ_{n1}) can be scattered in all sub-DAGs since there is no limitation for their resource allocation.

Algorithm 3 PrivacyModule(multiway-Cut)

```
1: procedure SCHEDULER($G(N, E)$)
2:     Let $terminalList$ be a new empty list
3:     $terminalList \leftarrow nodeContraction()$
4:     $sub - DAGList \leftarrow$ multiWayCut(terminalList)
5:     return the subDAGs
6: end procedure
```

The Multiterminal-Cut problem is proved to be MAX SNP-hard, even for small numbers such as $k = 3$, and many heuristics are already designed to solve it [40], [41]. Here, we use the model from [35] that is based on [40]. The outcome of Alg. 3 is one list of three sub-DAGs which becomes the input of the scheduler module.

5.1.2 BLP Method

The BLP model executes tasks on HCs by using predefined rules for deploying task and data. According to the BLP rules, all data and tasks must be deployed on HC instances with greater or equal privacy levels. The model in [42] automatically calculates all valid/feasible deployments of a workflow. Here, we have:

1) n_{i-j}: the network connecting platform i to platform j
2) $d_{i,x-j,y}$: the data sent from service i port x to service j port y
3) $l(z)$: the security location of z
4) $c(z)$: the clearance of z (the max l at which z may operate).

Then for each task n_i in the DAG, we add the following inequalities: The security level of the resource s_i on which the task is deployed must be greater than or equal to that of task n_i (Eq. 7).

$$l(s_i) \geq l(n_i) \tag{7}$$

For each edge (data connection) $d_{i,x-j,y}$ in the DAG, we add the following inequalities: The security level of the resource on which the service transmitting the data is deployed must be greater than or equal to that of the data (Eq. 8).

$$l(s_i) \geq l(d_{i,x-j,y}) \tag{8}$$

The security level of the resource on which the service receiving the data is deployed must be greater than or equal to that of the data (Eq. 9).

$$l(s_j) \geq l(d_{i,x-j,y}) \tag{9}$$

The security level of the network across which the data is transmitted must be greater than or equal to that of the data (Eq. 10).

$$l(n_{i-j}) \geq l(d_{i,x-j,y}) \tag{10}$$

Additionally to satisfy the clearance, for each task n_i and each edge (data connection) $d_{i,x-j,y}$ in the DAG, we add the following inequality:

$$c(n_i) \geq l(n_i) \tag{11}$$

where $c(n_i)$ represents the clearance of a task.

$$c(n_i) \geq l(d_{i,x-j,y}) \tag{12}$$

is the Bell-LaPadula "no read up" rule.

$$l(d_{i,x-j,y}) \geq l(n_i) \tag{13}$$

is the Bell-LaPadula "no write down" rule.

Applying all the rules above to a workflow, results in a list of valid deployment options where each deployment includes a list of sub-DAGs with individual privacy levels in the range of τ_{n1}, τ_{n2}, and τ_{n3}. Steps of computing the valid deployment options for a workflow using above inequalities are described in [42]. All the valid deployments build an individual input for the scheduling module.

5.2 Scheduling Module

The second essential section of our algorithm is the scheduling module; it assigns tasks of a workflow to instances in a HC without violating any task's privacy while meeting the workflow's deadline. To this end, we propose three policies; 1) MPHC-P1, 2) MPHC-P2, and 3) MPHC-P3. All policies implement an approximate schedule for tasks on a HC. All policies employ critical path (CP) and partial critical path (PCP) of sub-DAGs to produce the final schedule. PCP is defined according to two notions of *assigned node* and *critical parent*. An *assigned node* is a node that is already scheduled on a resource; a *critical parent* of a node n_i is an unassigned parent, n_p, with the latest data arrival time to n_i. *PCP* is defined as the critical parent (n_p) of n_i and the partial critical path of n_p if it has any unassigned parent. Details of these policies are described as follows:

5.2.1 MPHC-P1 (Single Path Policy)

The first scheduling policy is MPHC-P1; it takes a list of sub-DAGs from privacy preserving module and the main DAG with its deadline as inputs. In step 6 of Alg. 4, the partial critical path for each unassigned task is computed in each sub-DAG. The entire path is scheduled on the cheapest resource instance that can finish all its tasks before their latest finish time expires (step 7). An instance is suitable to execute a path when (1) the privacy level of the given path can be mapped to the selected resource, (2) the path can be scheduled on an already leased instance such that all tasks of the path are finished before their latest finish times, and (3) there are enough free slots in previously leased instances to execute all tasks of the given path before their latest finish times expires. If none of the conditions are met, a new instance from the cheapest applicable resource is launched –with regard to the privacy privilege of the path– to execute all its tasks before their latest finish time expires (Alg. 5).

Steps 6, 7, and 8 are repeated until all the tasks in the DAG are scheduled. Selecting the PCP in each sub-DAG instead of the main DAG leads to valuable advantages. First, there is no violation of privacy agreement because all tasks in a path are allocated on permitted instances. Second, because the length of a PCP in each sub-DAG is expected to be shorter –and usually is– than the PCP in the main DAG in most cases, the overall execution cost of workflows tends to be much lower.

5.2.2 MPHC-P2 (All Path Rank1)

The next scheduling policy, MPHC-P2, uses PCP Ranking; its details are shown in Alg. 6. Like before, inputs are: 1) the list of sub-DAGs from privacy preserving module, 2) the main DAG, and 3) the deadline.

The critical path (CP) and all PCPs of each sub-DAG are derived in step 6 of Alg. 6, and queued regarding their ranks in step 9. We rank all the retrieved PCs and PCPs using the path's sub-deadline. psd is the difference between the latest finish time of the last task $LFT(n_k)$, and the earliest start time of the first task $EST(n_1)$ in the path $p = \{n_1, n_2, ..., n_k\}$. The rank of a PCP is equal to the sub-deadline of the last task in that PCP (Eq. 14) [2].

$$\begin{aligned} &psd = LFT(n_k) - EST(n_1) \\ &subdeadline(n_i) = \frac{EFT(n_i) - EST(n_1)}{EFT(n_k) - EST(n_1)} \times psd \\ &PCPRank = subdeadline(n_k) \end{aligned} \tag{14}$$

Algorithm 4 SchedulingModule(MPHC-P1)

```
1: procedure MPH-PI( sub-DAGs List, $G(T, E), D$)
2:     add $n_{entry}$ and $n_{exit}$ to $G$
3:     compute $EST(n_i)$, $EFT(n_i)$ and $LFT(n_i)$ for all $n_i$
4:     for all Sub-DAGs do
5:         while unscheduled tasks exist do
6:             $path \leftarrow$ Find the PCP
7:             SchedulePath ($path$)
8:             Mark all tasks in the PCP as scheduled
9:         end while
10:    end for
11: end procedure
```

Algorithm 5 SchedulePath

```
 1: procedure SCHEDULEPATH($path$)
 2:     for all $ins$ in the InstanceList do
 3:         if $ins$ is not allowed then
 4:             continue
 5:         end if
 6:         if $ins$ is feasible then
 7:             save the ins if its cost is the minimum
 8:         end if
 9:     end for
10:     if any $ins$ was found then
11:         map all tasks in the $path$ to that $ins$
12:         exit
13:     end if
14:     for all $res$ in the ResourceList do
15:         find the best $res$ to schedule the $path$
16:     end for
17: end procedure
```

Algorithm 6 All Path Rank 1 (MPHC-P2)

```
 1: procedure MPH-PII( sub-DAGs List, $G(T, E), D$)
 2:     add $n_{entry}$ and $n_{exit}$ to $G$
 3:     compute all the $EST(n_i)$, $EFT(n_i)$ and $LFT(n_i)$
 4:     for all sub-DAGs do
 5:         add $n_{entry}$ and $n_{exit}$ to each sub-DAG
 6:         $path \leftarrow$ Find the PCPs/CPs
 7:         $pLevel \leftarrow$ the privacy level of the sub-DAG
 8:         Deadline Distribution of $path$
 9:         rank($path$)
10:     end for
11:     for all $path$ in the $queue$ do
12:         SchedulePath($path$)
13:     end for
14:     calculation()
15: end procedure
```

In step 12 each path is de-queued to be scheduled on the cheapest resource instance that can finish all tasks in the path before their latest finish times expires as also described in Alg. 5.

5.2.3 MPHC-P3 (All Path Rank2)

The only difference between this scheduling policy and MPHC-P2 is its ranking method. The CPs and PCPs are ranked based on the upward rank (HEFT [33]) of each task in a path. The rank of a PCP is the sum of all its tasks' upward rank (Eq. 15).

$$\begin{aligned} rank(n_{exit}) &= MET(n_{exit}) \\ rank(n_i) &= MET(n_i) + \max_{n_j \in n_i' sparents} \{rank(n_j) + TT(e_{ij})\} \\ PCPRank &= \sum_{all n_i \in DAG} rank(n_i) \end{aligned} \tag{15}$$

where $MET(n_i)$ is the Minimum Execution Time of task n_i on the HC's resource, n_j is one of the n_i's parents, and $TT(e_{ji})$ is the data transfer time between n_j and n_i. Each PCP is de-queued and scheduled on the cheapest resource instance that can finish all tasks in the path before their latest finish times using Alg. 5.

6 Evaluation Experiment

Our experimental setup is presented in this section. We compare MPHC-P1, MPHC-P2, and MPHC-P3 with another well-known algorithm, IC-PCPD2 [2], in this field. With the help of its original designers, we carefully modified IC-PCPD2 to preserve privacy, and thus it becomes a perfect match for a comparison.

6.1 Experimental Workflows

We selected two real-world workflow sample sets to gauge the efficiency of our MPHC's policies. The first set is from two real applications with actual privacy tags; medical research conducted in Newcastle University (Fig. 3) and a disease susceptibility workflow (Fig. 2). The second set is downloaded from the Pegasus Workflow Repository [43]. This set consists of hundreds of Montage, Epigenomics, LIGO, and Cybershake workflows, all available in DAX format (Directed Acyclic Graph in XML). Because privacy tags do not inherently exist in Pegasus workflows, we improvised and randomly –in a uniform manner– distributed them among their nodes.

6.2 Competitive Algorithm

The IC-PCPD2 algorithm [2] is a two-phase algorithm: 1) Deadline Distribution, and 2) Planning. In the deadline distribution phase, the overall deadline of the workflow is distributed over individual tasks, and in the planning phase each task is scheduled on an instance of a computation service according to its assigned sub-deadline. The deadline distribution phase tries to assign sub-deadlines to all unassigned parents of its input node, using PCP notion. The actual scheduling is carried out by the planning procedure which schedules each task on the cheapest instance that can execute the task before its sub-deadline. The sub-deadline of the entry task should be zero and the sub-deadline of the exit task is equal to the workflow's deadline in this algorithm .We modified the second phase to acquire only eligible HC instances by mapping tasks to authorized resources considering their privacy tag.

6.3 Experimental Setup

To estimate the effectiveness of our approach, we conducted a series of experiments on an Intel (R)-i7, 3.4GHz processors with 8G of RAM. We assumed a HC that offers different computation services (Table 2) which is inspired by Amazon EC2 pricing models [37]. Using VMware-vShpere [44], we built four clusters with 10 physical Intel Xeon servers. Each cluster had 100 VMs.

TABLE 2
Resources with hourly pricing model

ID	Price($ per hour in USD)	CPU(MIPS)	Privacy Tag
1	18.0	95	τ_{s2}
2	12.8	80	τ_{s2}
3	15.0	70	τ_{s1}
4	10.5	70	τ_{s3}
5	30.0	100	τ_{s1}
6	25.0	100	τ_{s3}

We divided VMs of the cluster into three categories considering the three privacy levels as shown in Table1. The average bandwidth between public-to-private and private-to-private VMs is set to 20Mbps (10Mbps upload and 10Mbps download) (similar to Amazon EC2 Standard) and 100Mbps, respectively.

6.4 Experimental Parameters

We considered the following parameters to create scenarios; experiments were run for each workflow sample under these metrics.

1) **DAG Size**: The size of real-world workflows (Montage, Epigenomics, LIGO, and Cybershake) are between 100-600. For the Healthcare workflow, the number of patients varies between 1 and 550; because each patient's workflow consists of 8 computing nodes in Fig. 3 and 14 nodes in Fig. 2, the DAG size ranged from 8 to 4400 and 13 to 7700 nodes for each healthcare workflow, respectively.
2) **Deadline Factor:** To assign a deadline to each workflow, we used a hypothetical scenario to calculate the fastest possible execution time of a workflow using the fastest computation service of a HC. All data transmission times are considered to be zero in that machine since it is negligible compared to the transmission time between two machines. The total running time ($TotalRunningTime$) of this schedule is considered as the lower bound for the running time of that workflow. Finally, to set a deadline for a workflow, we defined the deadline factor α (in range of 1 to 5) and used it to set the deadline of a workflow (Eq. 16). We believe/hypothesize that $\alpha = 1$ may lead to infeasible solutions since in the real world not all the fastest machines are available most of the times.

$$\begin{aligned} &Deadline\ Factor = \alpha \\ &Deadline = \alpha \times TotalRunningTime \end{aligned} \tag{16}$$

3) **Private Instance Limitation (PIL) Factor:** To run the experiments according to our scenario, we set a limitation (PIL factor) for the number of available private instances τ_{s2} and τ_{s3} to schedule workflows. To set the maximum number of private instances, for each workflow, we run both algorithms (MPHC-P1-3 and IC-PCPD2) without any resource restriction when $\alpha = 2$; the number of required VMs is called $TotalInsNo$. We defined three limitation factors β, $\beta1$, and $\beta2$ where $\beta1$ is the total number of available instances with τ_{s2} privacy tag, $\beta2$ is the total number of available instances with τ_{s3} privacy tag and $\beta = \beta1 + \beta2$. We ranged β from 0.3 to 1 to calculate the Number of Available Private Instances (NAPI) in the HC.

$$\begin{aligned} &PIL\ Factor = \beta \\ &NAPI = \beta \times TotalInsNo \end{aligned} \tag{17}$$

4) **Billing Cycle:** The billing cycle of most commercial cloud providers is based on the pay-as-you-go model. To study the effect of billing cycle to the overall quality of scheduling, we also evaluated the performance of MPHC-P1-3 and IC-PCPD2 for various billing time intervals (60 and 5 minutes).
5) **Total Cost:** We used Eq. 5 to measure the cost of running workflows with various deadlines and PIL factors. Since a large number of workflows with different attributes is used in this study, it is important to normalize the total cost of each workflow execution for better comparison [2]. To this end, we first calculate the "cheapest schedule" as scheduling all workflow tasks on a single instance of the cheapest computation service, according to their precedence constraints. Then, we can define the Normalized Cost (NC) of a workflow execution as follows Eq. 18:

$$NC = \frac{\text{Total Scheduling Cost}}{\text{Cheapest Scheduling Cost}} \tag{18}$$

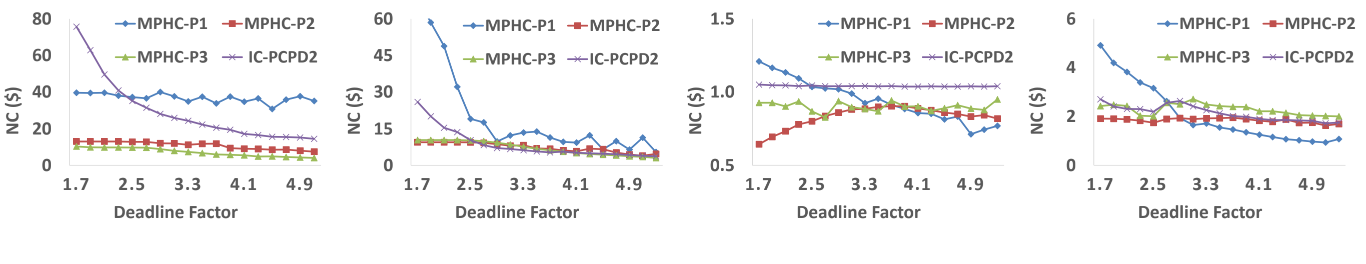
7 Results and Discussion

In this section, we present results of our augmented MPHC-P1-3 algorithm with its variations including detailed discussions.

7.1 Deadline Factor

Fig. 5 presents the cost of scheduling workflows with MPHC-P1-3 and IC-PCPD2 for DAGs with 300 nodes when $PIL = 0.5$ and the billing cycle is 60 minutes. For small deadline factors (tight deadlines) MPHC-P1 and IC-PCPD2 are the unprofitable option, whereas MPHC-P2 and MPHC-P3 are the less expensive policies for all workflows (for $\alpha < 3$). In Montage, MPHC-P3 is the best scheduler for various deadline factors. When there is less time limitation (loose deadlines), MPHC-P1 is a suitable scheduling policy for LIGO.

Fig. 6 shows the normalized cost of scheduling large workflows for all algorithms with the billing cycle of five minutes. The overall results are similar to the previous experiment except that the value of NC is decreased as expected. Because Epigenomics has tasks with long execution time, the value of the time interval doesn't have a considerable impact on NC and it is not decreased significantly. However, for Montage and CyberShake



(a) Montage (b) CyberShake (c) Epigenomics (d) LIGO

Fig. 5. Scheduling polices comparison with time interval equal to 60 min

workflows, tasks can efficiently utilize a larger fraction of short intervals due to the small execution times of their tasks. Therefore, the value of NC is considerably decreased. This is also the case for the LIGO workflow where the NC is decreased by about 50%.

MPHC-P1, P2, and P3 share the same path scheduling approach which means a series of dependent tasks are all scheduled on a single resource/VM; this is opposite to IC-PCPD2 where tasks are individually scheduled on resources. Therefore, in loose deadline scenarios, where time is not pressured, the modified IC-PCPD2 is more beneficial, since all tasks are individually scheduled on private/cheapest instances.

7.2 BLP VS Multiterminal-Cut

As mentioned, BLP is our alternative privacy preserving policy. We compare BLP and Multiterminal-Cut by scheduling our Medical Data Analysis workflow (Fig. 3) under MPHC-P2. In this experiment, the BLP policy results in 18 valid deployment options as shown in Table 3. The workflow Fig. 3 has 8 tasks $\{N_1, N_2, ..., N_8\}$ and in each deployment, calculated by BLP, tasks can be assigned to a resource with the selected privacy tag. For example in deployment option 1, task N_1 can be assigned to a resource with privacy τ_{s3}, task N_2 is mapped to a resource with privacy tag τ_{s3}, N_3 to a resource with privacy tag τ_{s3}, N_4 to a resource with privacy tag τ_{s2}, N_5 to a resource with privacy tag τ_{s3}, N_6 to a resource with privacy tag τ_{s3}, N_7 to a resource with privacy tag τ_{s1}, and N_8 to a resource with privacy tag τ_{s1}.

Despite the limitation in Multiterminal-Cut, it presents a lower scheduling cost when compared with the BLP method in Fig. 8; mainly because it considers edges weights while partitioning workflows. However, the scheduling cost of Run-17 and 14 of BLP is close to that of Multiterminal-Cut; this makes it a more favorable alternative, in case of difficulty in making tasks and their data privacy aligned for Multiterminal-Cut.

7.3 PIL Factor

The PIL factor should be taken into account to design an efficient scheduling broker using MPHC's variation. Experiments are run on workflows with 500 nodes with deadline factor of $\alpha = 2$. As shown in Fig. 7, the cost

TABLE 3
Valid Deployment Options for the Healthcare Data Analysis workflow

Dep	N_1	N_2	N_3	N_4	N_5	N_6	N_7	N_8
1	τ_{s3}	τ_{s3}	τ_{s3}	τ_{s2}	τ_{s3}	τ_{s3}	τ_{s1}	τ_{s1}
2	τ_{s3}	τ_{s3}	τ_{s3}	τ_{s2}	τ_{s3}	τ_{s3}	τ_{s1}	τ_{s2}
3	τ_{s3}	τ_{s3}	τ_{s3}	τ_{s2}	τ_{s3}	τ_{s3}	τ_{s1}	τ_{s3}
4	τ_{s3}	τ_{s3}	τ_{s3}	τ_{s2}	τ_{s3}	τ_{s3}	τ_{s2}	τ_{s1}
5	τ_{s3}	τ_{s3}	τ_{s3}	τ_{s2}	τ_{s3}	τ_{s3}	τ_{s2}	τ_{s2}
6	τ_{s3}	τ_{s3}	τ_{s3}	τ_{s2}	τ_{s3}	τ_{s3}	τ_{s2}	τ_{s3}
7	τ_{s3}	τ_{s3}	τ_{s3}	τ_{s2}	τ_{s3}	τ_{s3}	τ_{s3}	τ_{s1}
8	τ_{s3}	τ_{s3}	τ_{s3}	τ_{s2}	τ_{s3}	τ_{s3}	τ_{s3}	τ_{s2}
9	τ_{s3}	τ_{s3}	τ_{s3}	τ_{s2}	τ_{s3}	τ_{s3}	τ_{s3}	τ_{s3}
10	τ_{s3}	τ_{s3}	τ_{s3}	τ_{s3}	τ_{s3}	τ_{s3}	τ_{s1}	τ_{s1}
11	τ_{s3}	τ_{s3}	τ_{s3}	τ_{s3}	τ_{s3}	τ_{s3}	τ_{s1}	τ_{s2}
12	τ_{s3}	τ_{s3}	τ_{s3}	τ_{s3}	τ_{s3}	τ_{s3}	τ_{s1}	τ_{s3}
13	τ_{s3}	τ_{s3}	τ_{s3}	τ_{s3}	τ_{s3}	τ_{s3}	τ_{s2}	τ_{s1}
14	τ_{s3}	τ_{s3}	τ_{s3}	τ_{s3}	τ_{s3}	τ_{s3}	τ_{s2}	τ_{s2}
15	τ_{s3}	τ_{s3}	τ_{s3}	τ_{s3}	τ_{s3}	τ_{s3}	τ_{s2}	τ_{s3}
16	τ_{s3}	τ_{s3}	τ_{s3}	τ_{s3}	τ_{s3}	τ_{s3}	τ_{s3}	τ_{s1}
17	τ_{s3}	τ_{s3}	τ_{s3}	τ_{s3}	τ_{s3}	τ_{s3}	τ_{s3}	τ_{s2}
18	τ_{s3}	τ_{s3}	τ_{s3}	τ_{s3}	τ_{s3}	τ_{s3}	τ_{s3}	τ_{s3}

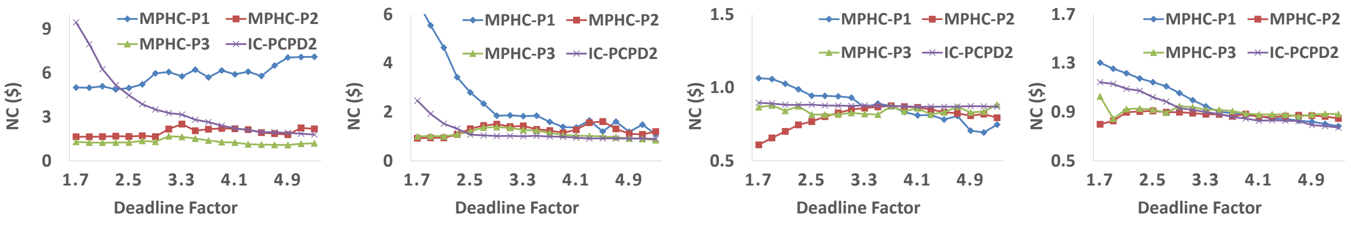
varies based on the cloud provider's policy. Apart from with LIGO, both MPHC-P2 and MPHC-P3 present a behavior that is independent of the number of private instances for the workflows. The exception is MPHC-P1, which is highly dependent on the number of available private instances.

7.4 Healthcare Workflows

We also run experiments for two healthcare workflows described in section 2 (Fig. 2 and Fig. 3). All three policies have the same behavior as for the other scientific workflows. Fig. 9 and Fig. 10 present the scheduling policies under various deadline factors for these two workflows. As we expected, MPHC-P2 and P3 indicate the most profitable policy among the rest policies in tide deadlines.

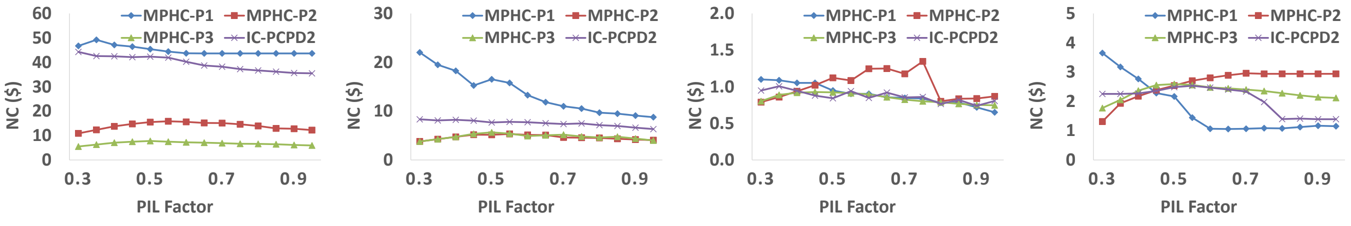
7.5 Privacy VS Time

We run an experiment to study the influence of preserving privacy on the total running time of a workflow. Although, deadline is one of the constraints of the problem and all our three suggested policies finished the workflow execution before the requested deadline, this experiment determines the fasted of four algorithms.



(a) Montage (b) CyberShake (c) Epigenomics (d) LIGO

Fig. 6. Scheduling polices comparison with time interval equal to 5 min



(a) Montage (b) CyberShake (c) Epigenomics (d) LIGO

Fig. 7. Scheduling polices comparison regarding PIL with time interval equal to 60 min for 500 tasks

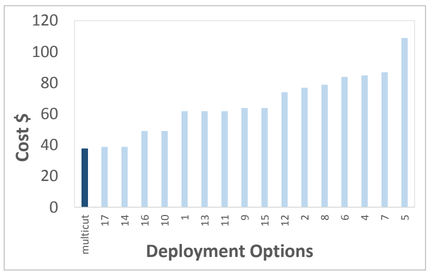


Fig. 8. Comparing the BLP VS Multiterminal-Cut

As it is presented in Table 4, MPHC-P1, P2, and P3 partition the workflow which leads to increase the data transfer time between different cloud instances. However, IC-PCPD2 assigns tasks individually to the eligible instances, therefore, it finishes the scheduling slightly faster than the other three policies.

TABLE 4
Impact of preserving privacy on the execution time

Policy	2	2.2	2.4	2.6	2.8	3	3.2	3.4
MPHC-P1	263	558	884	1236	1618	2034	2482	2961
MPHC-P2	195	406	652	946	1298	1705	2153	2632
MPHC-P3	196	406	639	903	1202	1528	1887	2279
IC-PCPD2	181	365	557	754	953	1154	1360	1570

7.6 Workflow Structure

Workflow structure has to be considered to obtain the optimal cost when scheduling workflows using MPHC-P1-3 policies. There are two factors in the workflow structure that have a direct impact on the scheduling cost. First is the number of private tasks in the workflow. To see the effect of the number of private tasks on the scheduling cost, we used the CyberShake workflow with 300 nodes under PIL Factor = 0.3. As Fig. 11 presents, increasing the number of private tasks in a workflow leads to a reduction in cost. This is due to the fact that private instances are set to have lower cost in our scenario. The second factor is the placement of tasks with equal privacy level in the workflow. If tasks with the same privacy level are linear (tasks have parent/child dependency), it commonly leads to longer PCPs in each sub-DAG. In contrast, when tasks with the same level of privacy are aligned parallel (tasks are siblings) there are more, but shorter, PCPs in each sub-DAG. As shown previously, MPHC-P1/2/3 policies use Multiterminal-Cut to partition the workflow regarding to privacy level of tasks. In a sub-DAG, a longer length PCP results in a length that is approximately equal to the length of the critical path in the main DAG. This causes a higher scheduling cost in scenarios where the workflow deadlines are tight. Therefore, structure of a workflow should be taken into account to acquire the optimal efficiency for time-critical scenarios, using MPHC-P2 and MPHC-P3 policies.

8 Conclusion

The classical problem of resource provisioning for workflow tasks moves into a new era with the introduction of federated clouds which combine private, community, and public clouds. This leads to more concerns about

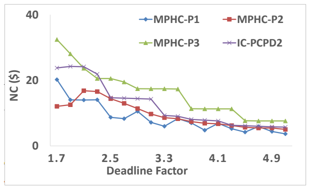


Fig. 9. Scheduling healthcare workflow in Fig.3

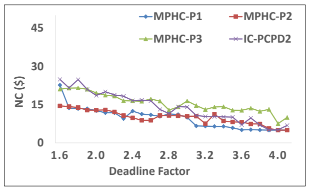


Fig. 10. Scheduling healthcare workflow in Fig.2

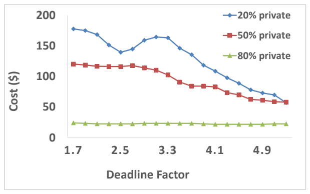


Fig. 11. Privacy distribution for CyberShake

satisfying the Service Level Agreements (SLAs) between clients and service providers. For example, the exposure of sensitive data and tasks in public clouds becomes a crucial barrier in workflow scheduling when several workflow tasks ought to be executed in off-premise cloud services in order to meet the desired deadline. Our objective in this study was to design and implement a workflow scheduling broker to minimize overall execution cost of workflows in HCs while satisfying concerns about their privacy and deadline. We contribute to this field by introducing MPHC algorithm including three policies (MPHC-P1/P2/P3) to minimize the workflow's execution cost under both task/data privacy and deadline constraints. In order to preserve the privacy, MPHC employs two different policies: the Multiterminal-Cut algorithm and BLP privacy model to partition a given workflow into sub-DAGs regarding its privacy levels for both tasks and data. In each sub-DAG, using the partial/critical paths, tasks of a workflow are assigned to different HC's instances/VMs before expiration of the desired deadline. We evaluate MPHC-P1/P2/P3 through comparing it with IC-PCPD2 as one of the well-known algorithms in this field. Results were promising and showed efficiency of our algorithm, MPHC, even in very time-pressured scenarios (deadline factor less than 3). For those time-pressured scenarios, we recommend MPHC-P2 and MPHC-P3 scheduling policies since they were successful in reducing the cost between 10- 20%. Whereas, for loose deadline scenarios, using modified IC-PCPD2 is more beneficial. However, all these assumptions are highly dependent on the workflow structure. In future work, we will therefore further explore the impact of workflow structure on selecting an efficient policy to schedule workflows on heterogeneous environments.

References

[1] D. Chou, "SOA And Cloud Computing."

[2] S. Abrishami, M. Naghibzadeh, and D. H. J. Epema, "Deadline-constrained workflow scheduling algorithms for Infrastructure as a Service Clouds," *Future Generation Computer Systems*, vol. 29, no. 1, pp. 158–169, Jan. 2013.

[3] L. F. Bittencourt and E. R. M. Madeira, "HCOC: a cost optimization algorithm for workflow scheduling in hybrid clouds," *Journal of Internet Services and Applications*, vol. 2, no. 3, pp. 207–227, Aug. 2011.

[4] R. Sakellariou, H. Zhao, E. Tsiakkouri, and M. D. Dikaiakos, "Scheduling workflows with budget constraints," in *Integrated Research in GRID Computing*. Springer, 2007, pp. 189–202.

[5] H. Liu, D. Xu, and H. Miao, "Ant colony optimization based service flow scheduling with various qos requirements in cloud computing," in *Software and Network Engineering (SSNE), 2011 First ACIS International Symposium on*. IEEE, 2011, pp. 53–58.

[6] M. Malawski, G. Juve, E. Deelman, and J. Nabrzyski, "Cost- and deadline-constrained provisioning for scientific workflow ensembles in IaaS clouds," *SC '12 Proceedings of the International Conference on High Performance Computing*, p. 22, Nov. 2012.

[7] A. Talukder, M. Kirley, and R. Buyya, "Multiobjective differential evolution for scheduling workflow applications on global grids," *Concurrency and Computation: Practice and Experience*, vol. 21, no. 13, pp. 1742–1756, 2009.

[8] R. Buyya, "Cost-Based Scheduling of Scientific Workflow Application on Utility Grids," in *First International Conference on e-Science and Grid Computing (e-Science'05)*. IEEE, pp. 140–147.

[9] L. Chunlin and L. Layuan, "Qos based resource scheduling by computational economy in computational grid," *Information Processing Letters*, vol. 98, no. 3, pp. 119–126, 2006.

[10] E.-K. Byun, Y.-S. Kee, J.-S. Kim, and S. Maeng, "Cost optimized provisioning of elastic resources for application workflows," *Future Generation Computer Systems*, vol. 27, no. 8, pp. 1011–1026, Oct. 2011.

[11] P. Watson, "A multi-level security model for partitioning workflows over federated clouds," *Journal of Cloud Computing*, vol. 1, no. 1, pp. 1–15, 2012.

[12] C. Zhang, E.-C. Chang, and R. H. Yap, "Tagged-mapreduce: A general framework for secure computing with mixed-sensitivity data on hybrid clouds," in *Cluster, Cloud and Grid Computing (CCGrid), 2014 14th IEEE/ACM International Symposium on*. IEEE, 2014, pp. 31–40.

[13] L. Zeng, B. Veeravalli, and X. Li, "SABA: A security-aware and budget-aware workflow scheduling strategy in clouds," *Journal of Parallel and Distributed Computing*, Sep. 2014.

[14] P. Mell and T. Grance, "The NIST Definition of Cloud Computing Recommendations of the National Institute of Standards and Technology," Tech. Rep.

[15] D.-K. Kang, S.-H. Kim, C.-H. Youn, and M. Chen, "Cost adaptive workflow scheduling in cloud computing," in *Proceedings of the 8th International Conference on Ubiquitous Information Management and Communication*. ACM, 2014, p. 65.

[16] S. Sharif, J. Taheri, A. Y. Zomaya, and S. Nepal, "MPHC: Preserving Privacy for Workflow Execution in Hybrid Clouds," pp. 272–280, Dec. 2013.

[17] S. Smanchat and K. Viriyapant, "Taxonomies of workflow scheduling problem and techniques in the cloud," *Future Generation Computer Systems*, vol. 52, pp. 1–12, 2015.

[18] J. Li, M. Qiu, Z. Ming, G. Quan, X. Qin, and Z. Gu, "Online optimization for scheduling preemptable tasks on iaas cloud systems," *Journal of Parallel and Distributed Computing*, vol. 72, no. 5, pp. 666–677, 2012.

[19] M. Bentounsi, S. Benbernou, C. S. Deme, and M. J. Atallah, "Anonyfrag," in *Proceedings of the 1st International Workshop on Cloud Intelligence - Cloud-I '12*. New York, New York, USA: ACM Press, Aug. 2012, pp. 1–8.

[20] S. B. Davidson, S. Khanna, S. Roy, J. Stoyanovich, V. Tannen, Y. Chen, and T. Milo, "Enabling privacy in provenance-aware workflow systems," 2011.
[21] X. Zhang, C. Liu, S. Nepal, and J. Chen, "An efficient quasi-identifier index based approach for privacy preservation over incremental data sets on cloud," *Journal of Computer and System Sciences*, vol. 79, no. 5, pp. 542–555, 2013.
[22] M. Ali, S. U. Khan, and A. V. Vasilakos, "Security in cloud computing: Opportunities and challenges," *Information Sciences*, vol. 305, pp. 357–383, 2015.
[23] Y. Li, W. Dai, Z. Ming, and M. Qiu, "Privacy protection for preventing data over-collection in smart city," 2015.
[24] Z. Yan, X. Li, M. Wang, and A. Vasilakos, "Flexible data access control based on trust and reputation in cloud computing."
[25] J. Zhou, Z. Cao, X. Dong, N. Xiong, and A. V. Vasilakos, "4s: A secure and privacy-preserving key management scheme for cloud-assisted wireless body area network in m-healthcare social networks," *Information Sciences*, vol. 314, pp. 255–276, 2015.
[26] K. Mivule, "Utilizing noise addition for data privacy, an overview," *arXiv preprint arXiv:1309.3958*, 2013.
[27] C. Dwork and R. Pottenger, "Toward practicing privacy," *Journal of the American Medical Informatics Association*, vol. 20, no. 1, pp. 102–108, 2013.
[28] C. Dwork, "Differential privacy: A survey of results," in *Theory and applications of models of computation*. Springer, 2008, pp. 1–19.
[29] M. Hardt, K. Ligett, and F. Mcsherry, "A simple and practical algorithm for differentially private data release," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 2339–2347.
[30] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," in *2011 Proceedings IEEE INFOCOM*. IEEE, Apr. 2011, pp. 829–837.
[31] I. Roy, S. T. Setty, A. Kilzer, V. Shmatikov, and E. Witchel, "Airavat: Security and privacy for mapreduce." in *NSDI*, vol. 10, 2010, pp. 297–312.
[32] J. Pei, J. Xu, Z. Wang, W. Wang, and K. Wang, "Maintaining K -Anonymity against Incremental Updates," no. Ssdbm, 2007.
[33] H. Topcuoglu and S. Hariri, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, no. 3, pp. 260–274, Mar. 2002.
[34] M. Malawski, G. Juve, E. Deelman, and J. Nabrzyski, "Cost-and deadline-constrained provisioning for scientific workflow ensembles in iaas clouds," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. IEEE Computer Society Press, 2012, p. 22.
[35] K. Deng, J. Song, K. Ren, D. Yuan, and J. Chen, "Graph-Cut Based Coscheduling Strategy Towards Efficient Execution of Scientific Workflows in Collaborative Cloud Environments," in *2011 IEEE/ACM 12th International Conference on Grid Computing*. IEEE, Sep. 2011, pp. 34–41.
[36] M. Tanaka and O. Tatebe, "Workflow Scheduling to Minimize Data Movement Using Multi-constraint Graph Partitioning," in *2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012)*. IEEE, May 2012, pp. 65–72.
[37] Amazon, "Amazon Elastic Compute Cloud (Amazon EC2)." [Online]. Available: http://aws.amazon.com/ec2/
[38] Microsft, "microsoft Azure." [Online]. Available: http://www.microsoft.com/azure/
[39] Z. Wen, J. Cala, and P. Watson, "A scalable method for partitioning workflows with security requirements over federated clouds," in *Cloud Computing Technology and Science (CloudCom), 2014 IEEE 6th International Conference on*. IEEE, 2014, pp. 122–129.
[40] E. Dahlhaus, D. S. Johnson, C. H. Papadimitriou, P. D. Seymour, and M. Yannakakis, "The Complexity of Multiterminal Cuts," *SIAM Journal on Computing*, vol. 23, pp. 864–894, Jul. 2006.
[41] M. Stoer and F. Wagner, "A simple min-cut algorithm," *Journal of the ACM*, vol. 44, no. 4, pp. 585–591, Jul. 1997.
[42] P. Watson and M. Little, "Multi-level security for deploying distributed applications on clouds, devices and things," in *Cloud Computing Technology and Science (CloudCom), 2014 IEEE 6th International Conference on*. IEEE, 2014, pp. 380–385.
[43] S. Bharathi, A. Chervenak, E. Deelman, G. Mehta, M.-H. Su, and K. Vahi, "Characterization of scientific workflows," in *2008 Third Workshop on Workflows in Support of Large-Scale Science*. IEEE, Nov. 2008, pp. 1–10.
[44] vmWare, "vmWare Clusters with vSphere." [Online]. Available: http://www.vmware.com/ap

Shaghayegh Sharif is a PhD student in Center of Distributed and High Performance Computing at the university of Sydney under supervision of professor Albert Zomaya. She received her master in the field of Computing Systems from KTH royal institute of technology (Sweden) in 2010. Her research interest is scheduling algorithms in heterogeneous environments such as clouds.

Paul Watson is Professor of Computer Science at Newcastle University, UK where he directs the Digital Institute. He graduated from Manchester University with a BSc in Computer Engineering (1983) and a PhD (1986). From 1990-5 he was a system designer at ICL. A Chartered Engineer and Fellow of the British Computer Society, he received the 2014 Jim Gray eScience Award.

Javid Taheri received his Bachelor and Masters of Electrical Engineering from Sharif University of Technology in 1998 and 2000, respectively. He received his Ph.D. in the field of Mobile Computing from the School of Information Technologies at the University of Sydney, Australia. He is currently working as Assosiate Professor in Department of Computer Science in Karlstad University, Sweden.

Surya Nepal received the BE and ME degree from the National Institute of Technology, Surat, India, and the Asian Institute of Technology, Bangkok, Thailand, respectively and the PhD degree from RMIT University, Australia then has been working in CSIRO, Australia. He is a principal research scientist working on services, trust and security aspects on cloud computing.

Albert Y. Zomaya is the Chair Professor of High Performance Computing & Networking in the School of Information Technologies, The University of Sydney. He served as the Editor in Chief of the IEEE Transactions on Computers (2011-2014). Professor Zomaya is the recipient of the IEEE Computer Society Technical Achievement Award (2014), the IEEE Technical Committee on Parallel Processing Outstanding Service Award (2011), the IEEE Technical Committee on Scalable Computing Medal for Excellence in Scalable Computing (2011). His research interests are in the areas of parallel and distributed computing and complex systems.