J. Parallel Distrib. Comput. 75 (2015) 141–151

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

SABA: A security-aware and budget-aware workflow scheduling strategy in clouds

Lingfang Zeng [a], Bharadwaj Veeravalli [a,*], Xiaorong Li [b]

[a] Department of Electrical and Computer Engineering, National University of Singapore, Singapore 117576, Singapore

[b] Institute of High Performance Computing, A*STAR, Singapore 138632, Singapore

HIGHLIGHTS

- In this paper we address the Workflow Scheduling in Clouds.
- We consider security and cost considerations.
- We propose a Security-Aware and Budget-Aware algorithm.
- We present rigorous theoretical analysis and verify on six different real datasets.
- We demonstrate a trade-off relationship between make span and the monetary cost.

ARTICLE INFO

Article history:
Received 30 October 2013
Received in revised form
23 July 2014
Accepted 2 September 2014
Available online 16 September 2014

Keywords:
Cloud computing
Workflow
Scheduling
Security
Budget

ABSTRACT

High quality of security service is increasingly critical for Cloud workflow applications. However, existing scheduling strategies for Cloud systems disregard security requirements of workflow applications and only consider CPU time neglecting other resources like memory, storage capacities. These resource competition could noticeably affect the computation time and monetary cost of both submitted tasks and their required security services. To address this issue, in this paper, we introduce immoveable dataset concept which constrains the movement of certain datasets due to security and cost considerations and propose a new scheduling model in the context of Cloud systems. Based on the concept, we propose a Security-Aware and Budget-Aware workflow scheduling strategy (SABA), which holds an economical distribution of tasks among the available CSPs (Cloud Service Providers) in the market, to provide customers with shorter makespan as well as security services. We conducted extensive simulation studies using six different workflows from real world applications as well as synthetic ones. Results indicate that the scheduling performance is affected by immoveable datasets in Clouds and the proposed scheduling strategy is highly effective under a wide spectrum of workflow applications.

1. Introduction

Cloud computing has been defined as a model for enabling convenient, on-demand network access to a shared pool of configurable resources (e.g., computing, networks, servers, storage, and applications) that can be rapidly provisioned and released with minimal management effort or Cloud provider interaction [29,16]. Users would like to pay a price which is commensurate the budget they can have.

Many of the features that make Cloud computing attractive, however, can also be problematic with traditional security models and controls. The security challenges sometime make Cloud computing formidable, especially for those Cloud service providers whose infrastructure and computational resources are owned by an outside party that sells those services to the general public. Security service in Cloud computing environment is extremely complex compared with that in a traditional data center. Many computational resources in a Cloud can result in a large attack surface [16]. Security is ranked the first as the greatest challenge attributed to Cloud computing. IDC (International Data Corporation) conducted a survey about Cloud service [14]: The number one concern about cloud services is security which is causing organizations to hesitate most when it comes to moving business workloads from *private Cloud* into *public Cloud*.

Workflow is one of most typical application models especially in scientific, engineering, and even business fields. Much work

* Corresponding author.
E-mail addresses: lingfangzeng@gmail.com (L. Zeng), jpdc@elsevier.com (B. Veeravalli), lixr@ihpc.a-star.edu.sg (X. Li).

http://dx.doi.org/10.1016/j.jpdc.2014.09.002
0743-7315/

have been conducted on workflow scheduling (e.g., [39,49,33,36,31,32,30]). However, most efforts have focused on the minimization of application completion time (makespan/schedule length) and/or time complexity. It is only recently that much attention has been paid to economic cost and security requirement in scheduling, particularly for Grids [45,44,15,35,43,37] and Clouds [8,22,5,48,28,1,23,41,17]. Workflows can be modeled as Directed Acyclic Graphs (DAGs) [12,13,20]. In these models, the existing approaches can address certain variants of the multi-criteria workflow scheduling problem, usually considering up to two contradicting criteria ([31,20,19] for example) being scheduled in some specific environments.

Some workflow applications may have to run in a distributed manner, because the required datasets are distributed, some with fixed locations. In these cases, data transfer is inevitable, and a data placement strategy would be needed to reduce the data transfer cost [10,11,47]. For example, if people *move* massive datasets, people have to pay the high cost of network transfer and storage (capacity). Nowadays, in some respects, data are users' *assets*, e.g. many scales and massive remote sensing data. These data assets can provide invaluable opportunities for business growth and profitability. So, data replication is not allowed. People make program (e.g. the operation of data copy/movement is forbidden) on data assets, ensure its safety (consequently be taken as *immoveable datasets* in this paper). Although, to protect data at rest and data in transit, people try to provide a trusted computing platform that prevents malicious software from taking control and compromising sensitive client and cloud application data [34]. Especially in multicloud computing environments, these security solutions cannot offer the same level of trust in terms of security guarantees and leave high value assets seriously exposed. Several other research works have studied data placement and task assignment for scientific workflows in clouds. Literature [46,47,9] proposed heuristics to optimize file access for scientific workflow in data centers.

Our proposed security-aware and budget-aware workflow scheduling scheme in this paper, referred to as, SABA, aims to minimize workflow execution time within user's security requirement and budget constraint in Cloud environments where multidimensional computing resources are considered. Our main contributions are summarized below:

- We present a security-aware and budget-aware workflow scheduling algorithm, called SABA, to provide customers with shorter makespan as well as secured scheduling under budget constraint. The results presented provide important guidelines for improving the security and scalability of practical applications.
- We investigate the Cloud workflow management with security implications and extend the scheduling model to multidimensional computing resources, such as computation, network bandwidth, memory and storage because the resource competition could noticeably affect the computation time and monetary cost of both submitted tasks and their required security services.
- We introduce immoveable dataset concept, which is simple but effective, for proposed scheduling model in the context of Cloud systems. We design efficient approaches to cluster tasks based on "data dependency". Note that the issue of clustering and prioritization in task scheduling, a focus of this paper, answers the most fundamental question, i.e., "which task in a workflow is assigned to the data center?". The proposed task clustering based on data dependency can directly reduce the time and the cost of accessing file data, especially in Cloud systems.
- We examine the proposed SABA scheme through extensive simulations and experiments on six real-world workflow applications. We examine the makespan as well as the speedup under various situations especially for communication-intensive workflows and those with wider degree of parallelism. Results demonstrate that our SABA scheme is highly effective and efficient in improving performance and cost of workflow scheduling, and can provide security and efficient service for task scheduling.

The remainder of this paper is organized as follows. We review the related work in Section 2. We describe the system model and schedule problem in Section 3. The proposed scheduling strategy is presented in Section 4. The performance evaluation approaches and simulation details and results are conducted in Section 5. We conclude this paper in Section 6.

2. Related work

A variety of budget-aware Cloud workflow scheduling methods have been proposed, e.g. BaTS [30], HCOC [5], PBTS [8], CCSH [22] and ScaleStar [48]. Zhu et al. [51] proposed a dynamic scheduling for fixed time-limit and resource budget constraint tasks. Literature [44,15] designed genetic algorithms to solve the scheduling problems considering the budget and deadline of entire network. Malawski et al. [27] provided cost- and deadline-constrained provisioning for scientific workflow for Cloud environments. Zheng et al. [50] considered a novel heuristic for the execution of the workflow which would allow providers to decide whether they can agree with the budget-deadline constraints set by the user.

However, most of well-known scheduling approaches ignore security issues, and only few groups of researchers investigate the security-driven scheduling domain from different angles in various contexts. Azzedin and Maheswaran [2] suggested integrating the trust concept into Grid resource management. They proposed a trust model that incorporates the security implications into scheduling algorithms. However, their trust model and scheduling algorithm are not suitable for Cloud. Song et al. [35] developed three risk-resilient strategies and a genetic algorithm based scheme (Space Time Genetic Algorithm, STGA) to provide security assurance in grid job scheduling. STGA considers only batch scheduling where jobs are independent of each other, and hence it is not suitable for parallel applications while precedence constraints and communications among tasks exist in a single application. Xie and Qin [43] studied a family of dynamic security-aware scheduling algorithms for homogeneous clusters and heterogeneous distributed systems. Their studies addressed the applications' demands for both real-time performance and security.

Mace et al. [26] provided general security solutions to choose what workflows, or subsets of workflows executed in a public cloud environment while ensuring the requirements of enterprise security and compliance. Watson [42] applied a rule-based Bell–LaPadula model to workflow security. Subjects and objects are assigned with different security levels. The directed graph of a workflow is used to analyze dependencies with multilevel security policies such as the No-read-up and No-write-down properties of the Bell–LaPadula [3] model. These methodologies could be valuable in defining security levels for services or application tasks and are orthogonal to the strategies described in this paper. Tang et al. [37] introduced task priority rank to estimate security overhead of tasks. They proposed a security-driven scheduling algorithm for DAGs which can achieve high quality of security for applications.

However, the risk and security models used in [35,43,37] are only for illustration purpose in traditional distributed computing environments. For Cloud applications, a more realistic security model should be proposed to approximately measure security offer and requirements. Moreover, several challenges need to be

Table 1
Major notations used in this paper.

Symbol	Definition
D_{read}	The disk read throughput of a single VM (Virtual Machine)
D_{write}	The disk write throughput of a single VM
B	The network bandwidth between data centers
α_x	The ratio of the temporary data which will be sent from VM v_x to other VM
θ	The input data transmitting security cost factor
r	The replication factor used for the task's output data. If no replication is used, $r = 1$
D_i	The total amount of input data for a given workflow
di_x	The amount of input data in VM v_x
D_m	The total amount of temporary data created by a given workflow
dm_x	The amount of temporary data in VM v_x
D_o	The total amount of output data for a given workflow
do_x	The amount of output data in VM v_x
$\tau_{i,j}$	The disk I/O time of a task t_i in a VM v_j
$\epsilon_{i,j}$	The amount of data to be transmitted from VM v_i to VM v_j
SR_i	The security level requirement of task t_i
sr_i^k	The required security range of task t_i for the kth security service
sp_j	The security services provided by VM v_j
SC^k	The security overhead of the kth security service
pr_i	The priority rank of task t_i
$ds_{i,k}$	The dataset transmission time from VM v_i to v_k

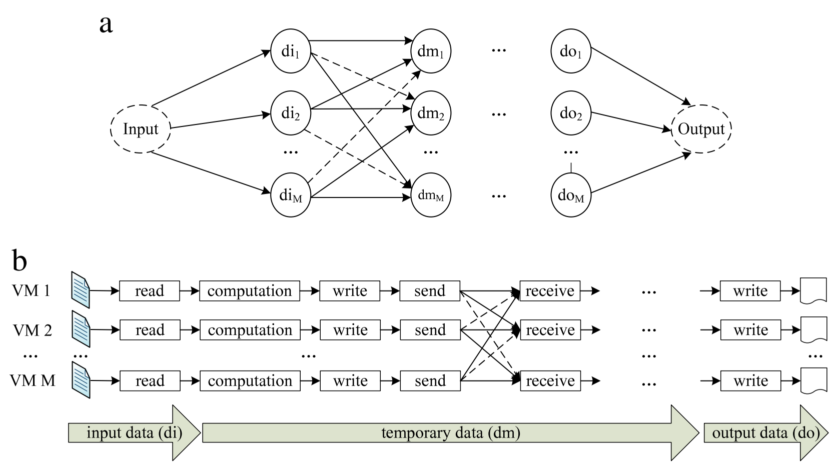


Fig. 1. The system model. (a) Data from a workflow arriving at the Cloud system can be assigned to M VMs with input data di, dm is the temporary data created by Cloud system, and do is the output data of Cloud system. (b) A map-reduce example.

addressed in this research topic in Cloud environments. Existing researches cannot be applied in Cloud for workflow applications with security requirements for three main reasons: Firstly, existing methods only consider CPU time which is one of the computing resources consumed by security services. Nonetheless, security services require other resources like memory, storage capacities, and network bandwidth. The submitted parallel tasks might compete for these resources. As a result, the resource competition could noticeably affect the computation time and monetary cost of both submitted tasks and their required security services. Secondly, current state of the arts considers all datasets are moveable. However, for real world workflow applications, the immoveable datasets (e.g. in private Cloud) must also be considered due to the security constraints. Thirdly, resources provided by CSPs (Cloud Service Providers) typically are shared by multi tenants who do not know each other. Sharing an infrastructure with unknown tenants can be a major risk for some workflow applications and requires a high level of assurance with the strength of the security mechanisms for logical separation.

In general, processing time, monetary cost, and security are three typical constraints for workflow execution on Cloud services. Given this motivation, we focus on developing a security-aware and budget-aware workflow scheduling strategy based on user's security requirements and budget constraints in Cloud environments.

3. System model and definitions

For ease of understanding, we summarize the notations and their meanings used throughout of this paper in Table 1.

In the proposed system, we follow the DAG model in [18] and extent its security level requirements to achieve security-aware workflow scheduling since the presented model is representative and commonly implemented in such systems. We assume that a workflow application is composed of a number of tasks and each of them may require input dataset(s) and output dataset(s). Our scheme schedules large number of such workflow tasks onto a given Cloud platform. The edges usually represent precedence constraints: each edge $e_{i,j} = \langle t_i, t_j \rangle$ represents a precedence constraint that indicates that task t_i should complete its execution before task t_j starts. $e_{i,j}$ represents the amount of inter-task communication involved. A task t_i has a weight ω_i ($1 \le i \le N$, N is the number of tasks in a workflow), corresponding to the execution time of task t_i on a baseline VM platform, and an edge $e_{i,j}$ ($1 \le i, j \le N$) has a weight $\epsilon_{i,j}$, which corresponds to the amount of data to be transmitted from VM v_i to VM v_j.

Our system model (Fig. 1) assumes that input data is distributed across all participating VMs in a Virtual Cluster (VC), and that each VM retrieves its initial input from local storage. For a VM v_j, the I/O (including disk read/write and network transmission) time of a

Table 2
Cryptographic overheads.

Algorithms	Overheads (μs)
AES encrypt (512 bits)	16
HMAC-SHA1 (512 bits)	13
AES encrypt (64 kB)	1,960
AES decrypt (64 kB)	1,835
HMAC-SHA1 (64 kB)	3,192
RSA signature (160 bits)	5,618
RSA verify (160 bits)	582
IBE signature (160 bits)	16,150
IBE verify (160 bits)	6,694

task t_i can be defined as Eq. (1).

$$\tau_{i,j} = \begin{cases} \dfrac{di_x + dm_x}{D_{read}} + \dfrac{dm_x + do_x}{D_{write}}, & \text{if } wb\&im; \\ \dfrac{di_x}{D_{read}} + \dfrac{dm_x + do_x}{D_{write}}, & \text{if } im; \\ \dfrac{di_x + dm_x}{D_{read}} + \dfrac{2dm_x + do_x}{D_{write}}, & \text{if } wb\&em; \\ \dfrac{di_x + dm_x}{D_{read}} + \dfrac{dm_x + do_x}{D_{write}}, & \text{if } em \end{cases} \tag{1}$$

where, "wb&im" denotes "write back and in-memory", "wb&em" denotes "write back and external memory". Eq. (1) takes the "write back" option (yes or no) and the required memory type (in-memory operation or external memory operation) into consideration.[1] Using the "wb&im" (write back and in-memory) as an example, the processing includes three disk operations (transfers) and one network operation (transfer): First, to execute computation, reading each input data from disk (di_x/D_{read}); Second, to flush data to the secondary storage, writing the temporary data to disk (the write to external storage media) (dm_x/D_{write}); Third, to get the temporary data from the secondary storage, reading the temporary data from external storage media (dm_x/D_{read}); Fourth, to persist data to external storage media, writing output data to external disk (do_x/D_{write}). Putting all of this together produces the equation shown. The amount of data to be transmitted from VM v_i to VM v_j can be defined as Eq. (2).

$$\epsilon_{i,j} = [\alpha_x \times dm_x + (r-1) \times do_x](1+\theta) \tag{2}$$

where θ is the input data transmitting security cost factor, $\theta > 0$, α_x denotes the ratio of the temporary data which will be sent from VM v_x to other VM, r denotes the replication factor used for the task's output data (r will be more than one if there are replication of the datasets; otherwise r is equal to 1). The equation shows that one output data copy (produced locally) and send $(r-1)$ replicated to other nodes.

We consider that each task may require a few security services (e.g. authentication, integrity, and confidentiality) with various security levels specified by the user. For example, $\mathbb{SR}_i$ is the set of security requirements of the task t_i. Security requirement SR_i of task t_i can be specified as a q-vector $SR_i = [sr_i^1, sr_i^2, \ldots, sr_i^k, \ldots, sr_i^q]$, where sr_i^k represents the required security level of the kth security service ($1 \le k \le q$). The security services provided by VM v_j is defined as sp_j. As security services introduce overhead to the existing computing systems [37], let SC^k be the security overhead of the kth security service (e.g. Table 2). Then, the security overhead $SC_{i,j}^k$ experienced by t_i on VM v_j can be calculated by using Eq. (3). The overall security overhead of t_i on VM v_j with security requirements for the above services is calculated as Eq. (4).

$$SC_{i,j}^k = \begin{cases} 0, & \text{if } sr_i^k \le sp_j^k; \\ \beta_j^k (sr_i^k - sp_j^k), & \text{otherwise} \end{cases} \tag{3}$$

where β_j^k is the security cost factor for kth security service of VM v_j and $\sum_{k=1}^{q} \beta_j^k = 1$.

$$SC_{i,j} = \sum_{k=1}^{q} \beta_j^k (sr_i^k - sp_j^k). \tag{4}$$

The priority rank of task t_i is calculated by adding the computation and I/O costs traversing the DAG upward from the exit task to task t_i. The priority rank is defined as Eq. (5).

$$pr_i = \frac{1}{V} \sum_{j=1}^{V} (w_{i,j} + \tau_{i,j} + SC_{i,j}) + \max_{t_k \in succ(t_i)} (ds_{i,k} + pr_k) \tag{5}$$

where $succ(t_i)$ is the set of successors of any task t_i. The value of pr_j is the priority rank of immediate successors of task t_i. The dataset transmission time ($ds_{i,k}$) is defined by Eq. (6).

$$ds_{i,k} = \begin{cases} \epsilon_{i,k} \cdot \lambda_{i,k}, & \text{if } v_i \text{ and } v_k \text{ in the same } DC; \\ \dfrac{\epsilon_{i,k}}{B}, & \text{otherwise} \end{cases} \tag{6}$$

where λ_{v_i,v_k} is the inverse of the bandwidth of the link between VM v_i and VM v_k, B is the network bandwidth between data centers. Since the priority rank is calculated by traversing the task graph upward, the priority rank of *exit* task is equal to:

$$pr_{exit} = \frac{1}{V} \sum_{j=1}^{V} (w_{exit,j} + \tau_{exit,j} + SC_{exit,j}). \tag{7}$$

The earliest start time $est(t_i, v_k)$ and earliest finish time $eft(t_i, v_k)$ of, a task t_i on a VM v_k are defined as:

$$est(t_i, v_k) = \begin{cases} 0, & \text{if } t_i = t_{entry}; \\ \max\{avail(k), \max\limits_{t_x \in pred(t_i)} \\ \quad (eft(t_x, v_m) + ds_{m,k})\}, & \text{otherwise} \end{cases} \tag{8}$$

$$eft(t_i, v_k) = est(t_i, v_k) + \omega_{i,k} + \tau_{i,k} + SC_{i,k} \tag{9}$$

where $pred(t_i)$ is the set of immediate predecessor tasks of task t_i, $avail(k)$ is the earliest time at which VM v_k is ready for task execution. v_k is the VM scheduled to task t_i, v_m is the VM scheduled to task t_x. If t_x is the last assigned task on VM v_k, then $avail(k)$ is the time that VM v_k completes the execution of the task t_x and it is ready to execute another task when we have a non-insertion-based scheduling policy. The inner max block in the *est* equation returns the *ready time*, i.e., the time when all data needed by t_i has arrived at VM v_k.

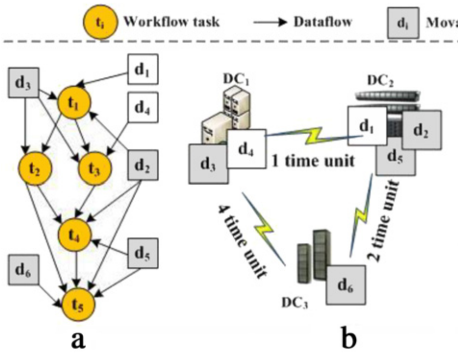
We consider heterogeneous cloud environments where each VM can have different computational power (e.g., the number of CPU cores and configurations) and pricing rates. Similarly, each data storage and network devices could have different storage capacity (in unit of GB) and network bandwidth associated with different pricing rates, respectively.

Let us say VM v_j takes $w_{i,j}$ time units to execute task t_i with price p_j, the computation monetary cost of running task t_i on VM v_j can be calculated as:

$$Cost_i = cost(t_i, v_j) = (w_{i,j} + \tau_{i,j} + SC_{i,j}) \cdot p_j \tag{10}$$

where p_j is the monetary cost per time unit to execute task t_i on VM v_j, $w_{i,j}$ is the execution time of a task t_i on the VM v_j.

[1] "Write back" denotes the write to the backing store is postponed until the cache blocks containing the data are about to be modified/replaced by new content. In-memory operation means the data fits in memory. External memory operation means the data does not fit in memory and an external memory is used, which involves writing data out to disk.



t_i	ω_i	DC_k	d_i	s_i	DC_k	$e_{i,j}$	$\varepsilon_{i,j}$
t_1	13	DC_2	d_1	20	DC_2	$e_{1,2}$	10
t_2	4		d_2	4		$e_{1,3}$	7
t_3	3	DC_1	d_3	2		$e_{2,4}$	9
t_4	6		d_4	25	DC_1	$e_{3,4}$	6
t_5	2		d_5	7		$e_{2,5}$	3
			d_6	3		$e_{4,5}$	5

c

Fig. 2. Example of workflow scheduling. (a) A simple workflow instance; (b) workflow data placement in 3 data centers; (c) workflow task computation time and workflow edge communication/data size; because some data are immoveable, these datasets belong to corresponding data center, and the same to associated task(s).

The total computation monetary cost of a workflow $Cost_{comp}$ can be calculated as:

$$Cost_{comp} = \sum_{j=1}^{V} \sum_{i=1}^{N} (w_{i,j} + \tau_{i,j} + SC_{i,j}) \cdot p_j \tag{11}$$

where N is the number of tasks in a workflow, $w_{i,j}$ is the execution time of task t_i on VM v_j, and V is the number of VMs.

In practice, if a VM is not released to CSPs, the user will be charged even if the VM is idle. Thus, Eq. (11) can be redefined as follows:

$$Cost_{comp} = \sum_{k=1}^{V} (eft_{t_j,v_k} - est_{t_i,v_k} \cdot p_k) \tag{12}$$

where eft_{t_j,v_k} is the earliest finish times of the last task t_j in the kth VM with price p_k, est_{t_i,v_k} is the earliest start times of the first task t_i in the kth VM, V is the number of VMs.

The total storage monetary cost of a workflow $Cost_{stor}$ can be calculated as:

$$Cost_{stor} = r \sum_{k=1}^{V} (di_k + do_k) \times CPC \tag{13}$$

where $r \sum_{k=1}^{V} (di_k + do_k)$ denotes the permanent storage required by the input and output data of a workflow, and CPC is the capacity monetary cost per GB of the storage.

For network transmission between data centers, the total monetary cost of the workflow $Cost_{tran}$ can be calculated as:

$$Cost_{tran} = \varphi \sum_{k=1}^{V} do_k \times CPB \tag{14}$$

where $\varphi \sum_{k=1}^{V} do_k$ denotes the amount of data transmitted, φ is the ratio of amount of data sent between different DCs to the total amount of data generated, and CPB is the bandwidth monetary cost per GB for data transfer.

The overall execution time of $\mathbb{G}$, or makespan, is defined as the time interval between the instant at which $\mathbb{G}$'s entry task starts its computation and the time instant at which $\mathbb{G}$'s exit task completes its computation. The makespan of the workflow $T_{makespan}$ can be calculated as:

$$T_{makespan} = \max\{eft(t_{exit})\} \tag{15}$$

where eft is defined in Eq. (9).

When a workflow is submitted to a Cloud, the scheduler allocates tasks to VMs in clouds. Our goal is to model and optimize Cloud services with the consideration of monetary cost and associated security constraints.

$$\text{minimize}(T_{makespan}), \text{ subject to,}$$
$$Bgt \geq Cost_{comp} + Cost_{tran} + Cost_{stor} + Cost_{spinup} \tag{16}$$

where Bgt is the user-given constraints on monetary cost, $Cost_{spinup}$ represents the monetary cost of system initialization. $Cost_{spinup}$ is also essential for user's economic constraints. Since they are not directly related to the workflow scheduling strategy, we do not give their details as the work presented here only focuses on the workflow tasks scheduling.

4. Proposed strategy: SABA

Our strategy consists of three phases. The first phase is clustering and prioritization, wherein a certain priority rank is calculated by Eq. (7) and assigned to each task in DAG. The second phase is the VM assignment, wherein each task (in order of its priority rank) is assigned to a VM that minimizes the cost function. These two phases (as a static assignment stage) are shown in *Algorithm* 1. The third phase is to overlap data movement and task execution during runtime stage (see *Algorithm* 2).

At the static assignment stage, the tasks are scheduled using their given execution times. This would be an initial solution to the scheduling problem, assuming there are no runtime changes for every task scheduled. However, in real-life situations, the execution times may vary during runtime. Hence, each scheduled task (but not executed) might need to be rescheduled according to the runtime resource performance. This is referred to as a dynamic case. At this runtime stage, the third phase adapts to the dynamic demands of the tasks and data movement.

4.1. Clustering and prioritization

The clustering is to build up initial clusters for the workflow tasks and datasets. Here we consider two types of datasets: *moveable datasets* and *immoveable datasets*. As the example shown in Fig. 2(a), datasets d_2, d_3, d_5 and d_6 are *moveable datasets* which can be moved (migrate/replicate) from one data center to another, while *immoveable datasets*, such as, d_1 and d_4, cannot be moved due to security or cost constraints.

We treat these two types of datasets with different policies: first, every *immoveable dataset* is clustered in accordance with its dependent data center. Second, every task with dependencies on any *immoveable dataset* is clustered in the same data center and is looked as an "immovable" task. Then, every input dataset of "immovable" task is clustered to its corresponding data center. Third, for all other tasks, if all the input datasets required by one task t_i are in the same data center, t_i is clustered to this data center. For example, in Fig. 2(a), $\{d_4, t_3, d_2, d_3, t_2\}$ is clustered to DC_1 and $\{d_1, t_1, d_3\}$ is clustered to DC_2.

4.2. VM assignment

We introduce an objective function referred to as *Comparative Factor* (CF). When a task t_i is assigned to a VM v_j with price p_j, we refer to this as an *assignment*. For a given task t_i, the *CF* value of an assignment of VM v_j (with p_j) with the best assignment of v' (with p') is defined as Eq. (17). A positive *CF* value indicates the finding of a new best scheduling.

Where $cost(t_i, v_j)$ and $cost(t_i, v')$ are calculated using Eq. (10) as the monetary cost of t_i on v_j (with p_j) and that of t_i on v' (with p'), respectively.

$$CF(t_i, v_j, v') = \begin{cases} \dfrac{\frac{eft(t_i,v_j)-eft(t_i,v')}{eft(t_i,v_j)-est(t_i,v_j)}}{\frac{|cost(t_i,v')-cost(t_i,v_j)|}{cost(t_i,v_j)}}, & \text{if } eft(t_i, v_j) > eft(t_i, v') \text{ and } cost(t_i, v') \neq cost(t_i, v_j); \\ 0, & \text{otherwise} \end{cases} \tag{17}$$

Similarly, $eft(t_i, v_j)$, $eft(t_i, v')$, $est(t_i, v_j)$, and $est(t_i, v')$ refer to the earliest finish/start times of the two task-VM allocations. For a given task, its *CF* value with each pair of VM and price are calculated using the current assignment of VM v' (with p'). Our algorithm selects the "task-to-VM" match which has the maximum *CF* value (steps from 9 to 15 in *Algorithm* 1).

```
1  Cluster immovable datasets and their dependent tasks;
2  Compute priority rank of any t_i by traversing graph upward;
3  Sort the tasks into a scheduling list in decreasing order by
   priority rank value;
4  foreach DC_x ∈ 𝔻ℂ do
5  |  Calculate initial available storage of all DCs;
6  end
7  for the scheduling list is not empty do
8  |  Remove the first task t_i from the scheduling list;
9  |  for any v_j ∈ DC_x do
10 |  |  for ∀t_i ∈ 𝕋 and ∀v' ∈ 𝕍 do
11 |  |  |  Compute CF(t_i, v_j, v') with v' using Eq. (17);
12 |  |  end
13 |  |  Select the maximum nonzero CF(t_i, v_j, v');
14 |  |  Replace v_j with v', assign t_i on v';
15 |  end
16 end
```

Algorithm 1: Static assignment stage of SABA.

4.3. Runtime stage: overlapped execution

At the runtime stage (shown in Algorithm 2), we have to design dynamic data movement with overlapped tasks' execution of workflows. Even though we know the tasks' correlations and related datasets that will be generated during these workflows' execution, it is not practical to move all required datasets to their target data centers. This is because there are large amount of different users. Each user may run one or more workflows which would have a large number of tasks and need very long time to complete. Furthermore, it is impractical and inefficient to reserve the computation power or storage space for future tasks or replicated datasets. This is because the tasks might not be scheduled until certain previous tasks are finished, wasting the reserved processor power and storage space during this time.

Assume ρ represents a percentage of a data center's total storage space, each data center will still have some storage available ($\rho_{threshold}$) to facilitate the overlapped execution. In the case that $\rho_{\max}$ is set to 100%, additional temporary storage space may need to be acquired to serve as a buffer before the computing process can be completed. In our system, for every data center, we reserve the runtime storage for generated datasets as $\rho_{threshold} = 35\%$[2] of the initial storage. So, in a data center, if the following inequality is satisfied, a task t_i can be executed:

```
Input: A set of DAG schedules 𝕊, e.g. a DAG
       𝔾 =< 𝕋, 𝔼, 𝔻_i, 𝔻_m, 𝕊ℝ > schedule S_x, S_x ⊂ 𝕊; A set of
       data centers 𝔻ℂ;
Output: All the tasks are finished;
while ∃S_x ∈ 𝕊 | S_x is not finished do
|  while ∃t_i ∈ 𝕋 | t_i is not dispatched for a S do
|  |  Select the first task t_i in the sorted 𝕋;
|  |  Replicate the required dataset(s) of task t_i from the
|  |  nearby DCs using resource state information; Remove
|  |  the dispatched tasks from 𝕋;
|  end
end
```

Algorithm 2: Runtime stage of SABA.

$$D_{t_i,gen} + D_{DC_j,used} \times (1 + \rho_{threshold}) < D_{DC_j,total} \tag{18}$$

where $D_{t_i,gen}$ is the size of the generated datasets of task t_i, $D_{DC_j,used}$ is the used storage space in the data center DC_j, and $D_{DC_j,total}$ is the total storage capacity of DC_j.

Because workflow scheduler may schedule multiple workflows at the same time, if Eq. (18) holds at $\rho_{threshold} = 35\%$, then the system can execute any other workflows. The system removes dataset when it is no longer used by other tasks. This strategy can improve workflow concurrency and allow efficient deadlock avoidance [40]. When tasks have been executed, new datasets are generated. The system will then decide where to put these datasets: either store them locally or allocate them to other data centers. In our work, the system will store the newly generated dataset in the local storage node first. At the same time, the dataflow scheduler periodically checks the system states and replicates the newly generated datasets to the data center where a task requires it as an input but the task itself depends some immovable dataset(s) and cannot be placed in other data centers.

5. Performance evaluation

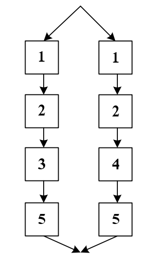
In this section, we compare the performance of SABA strategy with a well-known data placement strategy proposed by Yuan et al. [46] (denoted as "Dataplace"). We modify *Dataplace* strategy by considering the security level of a task and the immovable/movable datasets to enhance it and make fair comparison. We study the performance of both SABA and the modified *Dataplace* using six real world workflow applications as well as a wide spectrum of synthetic workflows.

5.1. Evaluation methodology

We set 4 data centers equipped with computation and storage resources. Since the storage capacity is bounded, there are cases where a data center does not have enough storage capacity needed to store a required dataset. In such case, one or more of the oldest and unused datasets are deleted until the new dataset can be stored. Other factor that would affect the performance is the number of immoveable datasets. We randomly choose some percentage of datasets from the existing data as immoveable datasets and randomly select some data centers for them. Also, we assume none of the tasks require *immoveable* input datasets from *different* data centers.

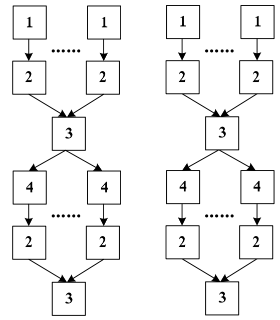
We employed the proposed schemes and the required Cloud entities (e.g., compute, bandwidth, and storage resources) in a simulator where parallel application graphs are based on some of the real world workflow applications, namely Gene2Life, LIGO, SIPHT,

[2] Mark Levin [21] shows that the percentage of storage managed by SANs is nearly the same for UNIX (65%) and Windows environments (63.6%).



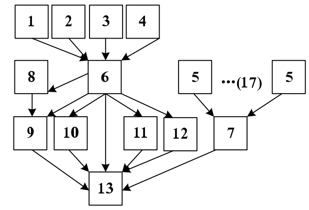
ID	Task	CT(s)	Input	Output
1	blast	180	0.1MB	1MB
2	clustalw	300	0.1MB	0.1MB
3	dnapars	30	4KB	4KB
4	protpars	30	4KB	4KB
5	drawgram	30	4KB	35KB

(a) Gene2Life.



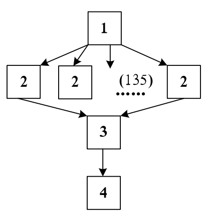
ID	Task	CT(s)	Input	Output
1	TmpltBank (34)	19	70MB	1MB
2	Inspiral (76)	460	75MB	300KB
3	Thinca (14)	6	16MB	30KB
4	TrigBank (42)	6	14KB	10KB

(b) LIGO.



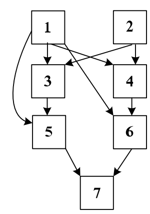
ID	Task	CT(s)	Input	Output
1	Transterm	33	4.65MB	435KB
2	Findterm	975	5MB	46MB
3	RNAMotif	44	5KB	1MB
4	Blast	3331	270MB	5MB
5	Patser (17)	2	4.3MB	10KB
6	SRNA	307	23MB	5MB
7	Patser_conate	1	120KB	120KB
8	FFN_parse	2	5MB	0.83MB
9	Blast_synteny	33	4MB	1.35MB
10	Blast_candidate	6	2MB	12KB
11	Blast_QRNA	13445	259MB	5MB
12	Blast_paralogues	5	1MB	1.24MB
13	SRNA_annotate	2	4MB	1.1MB

(c) SIPHT.



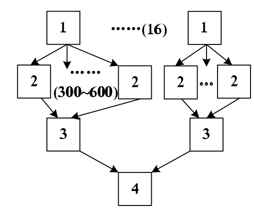
ID	Task	CT(s)	Input	Output
1	Pre Interproscan	30	13MB	100KB
2	Interproscan	5400	100KB	500KB
3	Post Interproscan	60	67.5MB	599MB
4	Motif	3600(256CPUs)	71MB	1432MB

(d) Motif.



ID	Task	CT(hr)	Input	Output
1	Adcirc	11(256CPUs)	534MB	21527MB
2	Wave Watch III	1(256CPUs)	810MB	2766MB
3	SWAN Outer South	8(10CPUs)	24293MB	14MB
4	SWAN Outer North	13(8CPUs)	24327MB	16338MB
5	SWAN Inner South	3(192CPUs)	21575MB	4.4GB
6	SWAN Inner North	4(160CPUs)	6383.42MB	3.8GB
7	Adcirc	.5(256CPUs)	8.734GB	6510MB

(e) NCFS.



ID	Task	CT	Input	Output
1	ColdDB&LoadDB	5mins	100MB	100MB
2	Merge DB	3hrs	2TB+100MB	~2TB
3	Validate Merge	1min	~2TB	~2.03TB
4	Update Production DB	1hr	~2.03TB	N/A

(f) PSMerge.

Fig. 3. Workflow structures of six applications with the information on computation time (CT) and input/output data size of each task [4,33].

Motif, NCFS, and PSMerge [33,4]. Note that the computation time and the data size of the six DAGs (shown in Fig. 3) are representative. We can reconfigure them according to the problem size while the structure remains the same.

As shown in Fig. 3, Gene2Life workflow [33] takes an input DNA sequence, discovers genes that match the sequence. This workflow has two parallel sequences. The results of the searches are parsed to determine the number of identified sequences that satisfy the selection criteria. The outputs trigger the launch of *clustalw*, a bioinformatics application that is used for the global alignment process to identify relationships. These outputs are then passed through parsimony programs for analysis. The two applications that may be available for such analysis are *dnapars* and *protpars* [33]. In the last step of the workflow plots are generated to visualize the relationships, using an application called *drawgram*.

The LIGO Inspiral Analysis Workflow [7] is used by the Laser Interferometer Gravitational Wave Observatory to detect gravitational waves in the universe. The detected events are divided into smaller blocks and checked.

SIPHT [24] automates the search for SRNA encoding-genes for all bacterial replicons in the National Center for Biotechnology Information database. SIPHT is composed of a variety of ordered individual programs on data.

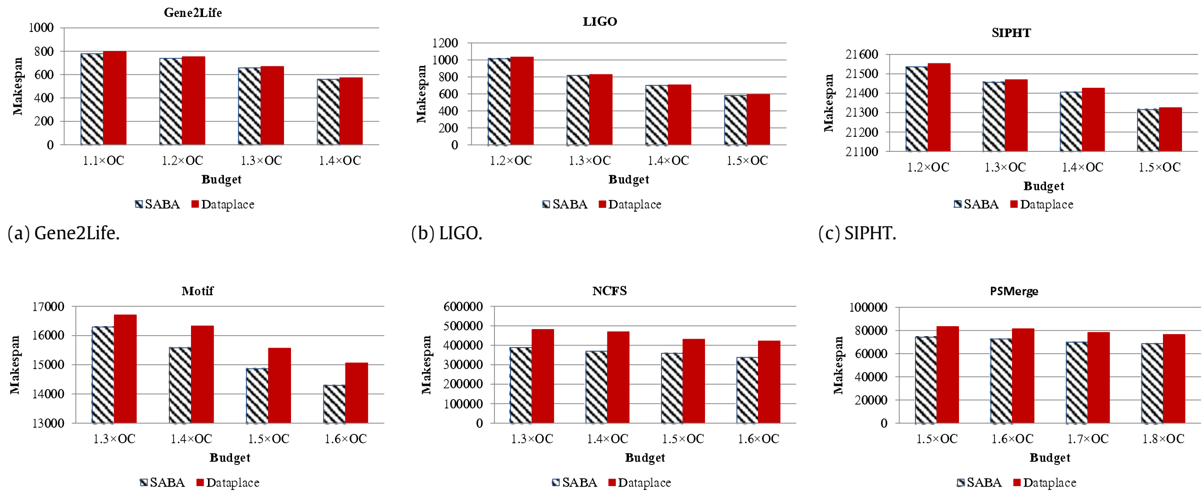
The Motif workflow [38] is computationally intensive. The first stage of the workflow assembles input data and processes the data that is then fed into *InterProScan* service. The executions of InterProScan are handled through Taverna and scripts. The motif workflow has a parallel split and merge paradigm where preprocessing spawns a set of parallel tasks that operate on subsets of the data. Finally, the results from the parallel tasks are merged and fed into the multi-processor application.

NCFS workflow [6] is focused on developing accurate simulation of storm surges in the coastal areas of North Carolina. The deployed system consists of a four-model system that consists of the Hurricane Boundary Layer (HBL) model for winds, Wave Watch III and SWAN for ocean and near-shore wind waves, and *Adcirc* for storm surge. The models require good coverage of the parameter space describing tropical storm characteristics in a given region for accurate flood plain mapping and analysis. Computational and storage requirements for these workflows are fairly large requiring careful resource planning. An instance of this workflow is expected to run for over a day.

PSMerge workflow [25] is data intensive. The workflows have a high degree of parallelism through substantial partitioning of data into small subsets.

Further information on these workflows is available in [33,4].

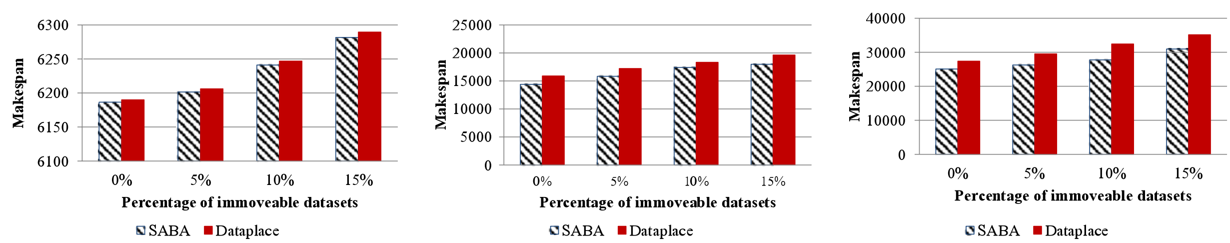
Since the aforementioned six workflows discussed may not cover the characteristics of all workflow applications, we evaluate our strategy rigorously using synthetic workflows of various structures. We generate a large number of workflows with parameters, such as the number of tasks N, the number of edges, and computing time using the procedure described in [37]. The *parallelism factor* (α), reflects the degree of parallelism of a random generated DAG and *Communication to Computation Ratios* (CCRs) is defined as



(a) Gene2Life. (b) LIGO. (c) SIPHT.

(d) Motif. (e) NCFS. (f) PSMerge.

Fig. 4. Comparison of the makespan of SABA and *Dataplace*. Tasks' ID in {} means they are in the same data center. (a) {1,2}, {3}, {4}, {5}; (b) {1}, {2}, {3}, {4}; (c) {1,2,3,4,6,8,9,10,12}, {11,13}, {5,7}; (d) {1}, {2}, {3}, {4}; (e) {1,3,5}, {2,4,6}, {7}; (f) {1,2}, {3,4}.



(a) Low communication (CCR = 0.1). (b) Medium communication (CCR = 1.0). (c) High communication (CCR = 2.0).

Fig. 5. Comparison of the *makespan* of SABA and *Dataplace* for synthetic workflows while the budget is: (a) $1.1 \times OC$; (b) $1.3 \times OC$; (c) $1.5 \times OC$.

the ratio of average communication cost to the average computation cost of a DAG [39]. Every set of the above parameters is used to generate several random graphs in order to avoid scattering effects. The results presented are the average of the results obtained for these graphs (average of 100 random workflows with 30,000 tasks).

We evaluate the performance in terms of the *makespan* (see Eq. (15)), and the *speedup* (see Eq. (19)) that is calculated by the sequential execution time (i.e., cumulative computation times, I/O times, and security overheads of the tasks in the graph) over the parallel execution time (i.e., the makespan of the output schedule). The sequential execution time is derived by assigning all tasks to a single processor that minimizes the cumulative time of the computation times, I/O times, and security overheads.

$$Speedup = \frac{\min\limits_{v_i \in \mathbb{V}} \left\{ \sum\limits_{t_i \in \mathbb{T}} (w_{i,j} + \tau_{i,j} + SC_{i,j}) \right\}}{makespan}. \quad (19)$$

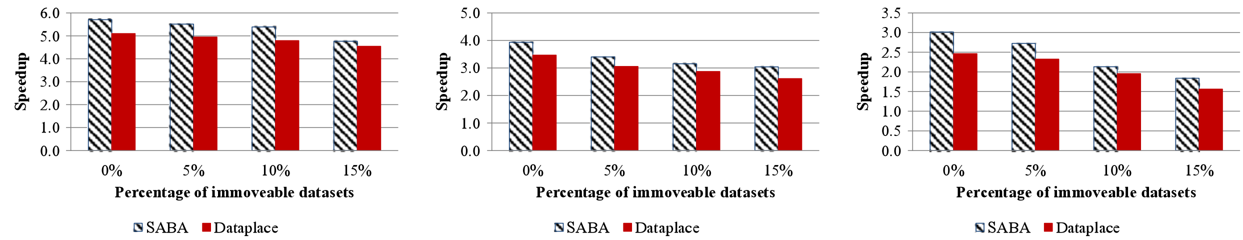
We use $OC = \sum_{i=1}^{N} (et_{t_i} \cdot count(t_i) \cdot p_h)$ to compute the Optimum monetary Cost (OC). The monetary cost is proportional to the resource time used by a workflow. et_{t_i} is the execution time of a task t_i, typically, the execution of a task consists of three phases, downloading of input data from the storage system, running the task, and transmitting output data to the storage system; $count(t_i)$ is the number of required type of physical host h for task t_i (e.g. a parallel processing task), p_h is the monetary cost per second of host h. This optimal monetary cost is the lower bound of the monetary cost of a workflow schedule. In the performance study, we use it to examine the performance of different algorithms under various budget constraints.

5.2. Experimental results

Fig. 4 compares the performance of proposed SABA with the modified "Dataplace". We apply SABA and *Dataplace* to the DAG files of six real world workflows and calculate the makespan of two schemes. The results show that SABA is better than *Dataplace* in most cases. SABA is noticeably better than *Dataplace* for Motif, NCFS and PSMerge which are higher communication workflows. There are remarkable reduction of the makespan by SABA under various budget constraints for the Motif workflows whose degree of parallelism varies widely. In addition, SABA is comparable to or slightly better than *Dataplace* for Gene2Life, LIGO, and SIPHT. In these experiments, SABA outputs *Dataplace* in most cases. It is because, although *Dataplace* strategy is intended to schedule tasks with security requirements, it does not optimize the quality of security by considering multiple competing resources.

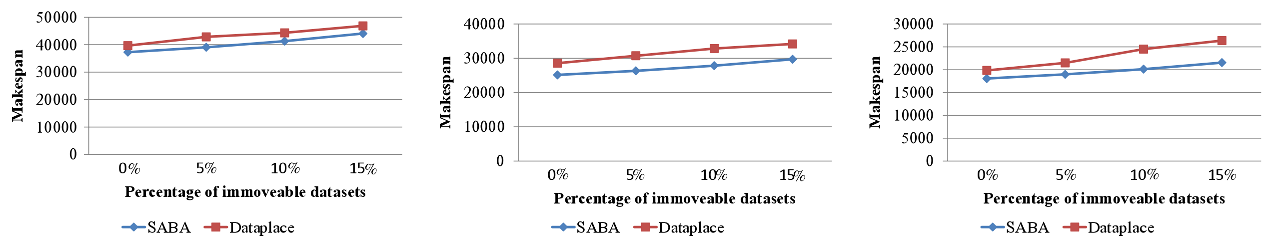
Fig. 5 compares the performance of SABA and Dataplace for synthetic workflows. The makespan produced by SABA and *Dataplace* strategies for various Communication to Computation Ratio (CCR) values are illustrated in Fig. 5. The SABA strategy is shorter than *Dataplace* strategy by: (0.06%, 0.08%, 0.09%, 0.13%), (10.46%, 8.82%, 5.27%, 9.27%), and (9.28%, 12.5%, 16.88%, 13.59%), for CCR is 0.1, 1.0 and 2.0, respectively. The first value of each parenthesized pair is the improvement achieved by SABA strategy over *Dataplace* strategy for no immoveable datasets, while the other three values are the improvement of SABA over *Dataplace* strategy for the workflows including 5%, 10% and 15% *immoveable* datasets, respectively.

The average makespan value of SABA strategy is shorter than *Dataplace* strategy by: 0.08%, 8.36% and 13.17%, for CCR is 0.1, 1.0 and 2.0, respectively. We observed that SABA strategy is more



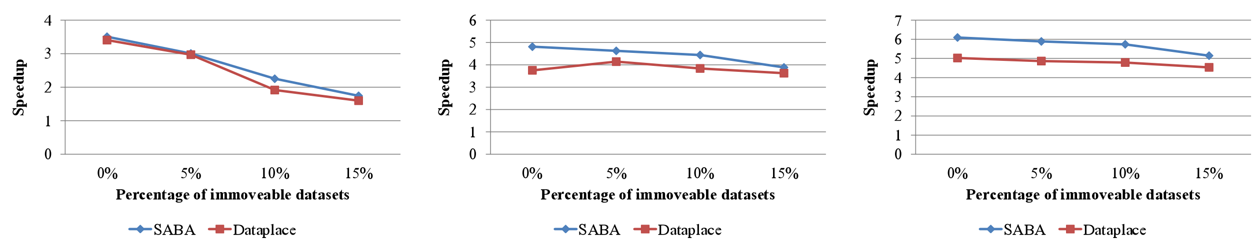
(a) Low communication (CCR $= 0.1$). (b) Medium communication (CCR $= 1.0$). (c) High communication (CCR $= 2.0$).

Fig. 6. Comparison of the *speedup* of SABA and *Dataplace* for synthetic workflows while the budget is: (a) $1.1 \times OC$; (b) $1.3 \times OC$; (c) $1.5 \times OC$.



(a) Low parallelism ($\alpha = 0.5$). (b) Medium parallelism ($\alpha = 2.0$). (c) High parallelism ($\alpha = 5.0$).

Fig. 7. Comparison of the *makespan* of SABA and *Dataplace* for synthetic workflows while the budget is: (a) $1.4 \times OC$; (b) $1.2 \times OC$; (c) $1.1 \times OC$.



(a) Low parallelism ($\alpha = 0.5$). (b) Medium parallelism ($\alpha = 2.0$). (c) High parallelism ($\alpha = 5.0$).

Fig. 8. Comparison of the *speedup* of SABA and *Dataplace* for synthetic workflows while the budget is: (a) $1.4 \times OC$; (b) $1.2 \times OC$; (c) $1.1 \times OC$.

suitable for those security-sensitive workflows with high percentage of immoveable datasets. In Fig. 5(c), SABA significantly outperforms the *Dataplace* strategy in terms of the makespan. Performance improvement of SABA over *Dataplace* is due to the fact that SABA assigns a task to a service provider VM not only considering its computational and I/O time but also its security demands. However, *Dataplace* schedules tasks only considering computational time and data movement, and the success of task execution inevitable causes the security overhead on Clouds. Thus, the makespan of *Dataplace* is longer than SABA.

The speedup values achieved by the two strategies with respect to certain CCR values are illustrated in Fig. 6. The average speedup value of SABA strategy is higher than those returned by *Dataplace* strategy by: 9.3%, 11.24%, and 14.42%, when the CCR is equal to: 0.1, 1.0 and 2.0, respectively.

We vary α from 0.5 to 5, to examine the performance sensitivity for the two strategies to the parallelism factor α. As shown in Fig. 7, the SABA strategy is shorter than *Dataplace* strategy by: (6.33%, 9.68%, 7.31%, 6.32%), (13.69%, 16.71%, 17.94%, 15.04%), and (9.86%, 13.28%, 21.76%, 22.39%), for α 0.5, 2.0 and 5.0, respectively. The first value of each parenthesized pair is the improvement achieved by SABA strategy over *Dataplace* strategy for no immoveable datasets, while the other three values are the improvement of SABA over *Dataplace* strategy for the workflows including 5%, 10% and 15% *immoveable* datasets, respectively. The average makespan value of SABA strategy is shorter than *Dataplace* strategy by: 7.38%, 15.87% and 17.14%, for α 0.5, 2.0 and 5.0, respectively. As the parallelism factor α increases, the improvement becomes more significant. The improvement of the speedup could be observed from Fig. 8.

6. Conclusion

Cloud redefines the security issues targeted on customer's workflow schedule. It is critical to investigate the Cloud workflow management with security implications and extend scheduling model to multidimensional computing resources, such as computation, network bandwidth, memory and storage. This work is one of the first attempts to address the security and budget awareness into task scheduling in Clouds. In this work we observed that the resource competition could noticeably affect the computation time and monetary cost of both submitted tasks and their required security services. Towards this end, we had proposed a security-aware and budget-aware workflow scheduling strategy (referred to as SABA). Our extensive simulation studies reveal that the proposed SABA shows considerable improvement under various situations especially for communication-intensive workflows and those with wider degree of parallelism. Furthermore, our schemes are able to deal with various types of datasets with security and budget requirements and improve resource utilization effectively by handling dynamic data transmission and execution at the same time.

Acknowledgments

This work is supported by A*STAR SERC, Singapore, under grant project "Reliable and Adaptive MapReduce-Enabled Cloud

Platform for Workflow Data Analytics Services (DVCS)", 102-158-0036. We thank all the anonymous reviewers whose comments noticeably improved the quality of this paper.

References

[1] V. Aravinthan, W. Jewell, Optimized maintenance scheduling for budget-constrained distribution utility, IEEE Trans. Smart Grid 4 (4) (2013) 2328–2338.

[2] F. Azzedin, M. Muthucumaru, A trust brokering system and its application to resource management in public-resource grids, in: Proc. of 18th International Parallel and Distributed Processing Symposium, IPDPS, 2004.

[3] D. Bell, L. LaPadula, Secure computer systems: mathematical foundations, Tech. Rep. 2547, Mitre Corp., Bedford, MA, March, 1973.

[4] S. Bharathi, A. Chervenak, E. Deelman, G. Mehta, M. Su, K. Vahi, Characterization of scientific workflows, in: Proc. of the 3rd Workshop on Workflows in Support of Large-Scale Science, 2008.

[5] L.F. Bittencourt, E.R.M. Madeira, HCOC: a cost optimization algorithm for workflow scheduling in hybrid clouds, J. Internet Serv. Appl. 2 (3) (2011) 207–227.

[6] B. Blanton, H. Lander, R. Luettich, M. Reed, K. Gamiel, K. Galluppi, Computational aspects of storm surge simulation, in: Ocean Sciences Meeting, 2008.

[7] D. Brown, A.D.P. Brady, J. Cao, B. Johnson, J. McNabb, Workflows for e-Science, in: A case study on the use of workflow technologies for scientific analysis: gravitational wave data analysis, Springer, 2006 (Chapter).

[8] E.-K. Byuna, Y.-S. Keeb, J.-S. Kimc, S. Maeng, Cost optimized provisioning of elastic resources for application workflows, Future Gener. Comput. Syst. 27 (8) (2011) 1011–1026.

[9] U.V. Catalyurek, K. Kaya, B. Ucar, Integrated data placement and task assignment for scientific workflows in clouds, in: Proc. of the Fourth International Workshop on Data-Intensive Distributed Computing, DIDC, 2011.

[10] A. Chervenak, E. Deelman, I. Foster, L. Guy, W. Hoschek, A. Iamnitchi, C. Kesselman, P. Kunszt, M. Ripeanu, B. Schwartzkopf, H. Stockinger, K. Stockinger, B. Tierney, Giggle: a framework for constructing scalable replica location services, in: Proc. of the ACM/IEEE Conference on Supercomputing, SC, 2002, pp. 1–17.

[11] A. Chervenak, E. Deelman, M. Livny, M.-H. Su, R. Schuler, S. Bharathi, G. Mehta, K. Vahi, Data placement for scientific applications in distributed environments, in: Proc. of the 8th Grid Computing Conference, 2007, pp. 267–274.

[12] J. Ding, Y. Wang, J. Le, Y. Jin, Dynamic scheduling for workflow applications over virtualized optical networks, in: IEEE Conference on Computer Communications Workshops, 2011, pp. 127–132.

[13] P.-F. Dutot, T. N'Takpe, F. Suter, H. Casanova, Scheduling parallel task graphs on (almost) homogeneous multicluster platforms, IEEE Trans. Parallel Distrib. Syst. 20 (7) (2009) 940–952.

[14] F. Gens, IT Cloud services user survey, pt.2: top benefits & challenges, October 2008. URL: http://blogs.idc.com/ie/?p=210.

[15] G. Gharooni-fard, F. Moein-darbari, H. Deldari, A. Morvaridi, Scheduling of scientific workflows using a chaos-genetic algorithm, in: International Conference on Computational Science, ICCS, Vol. 1 (1), 2010, pp. 1445–1454.

[16] W. Jansen, T. Grance, Guidelines on security and privacy in public cloud computing, January 2011. URL: http://csrc.nist.gov/publications/drafts/800-144/Draft-SP-800-144cloudcomputing.pdf.

[17] D. Jung, J. Lim, H. Yu, J. Gil, E. Lee, A workflow scheduling technique for task distribution in spot instance-based cloud, in: Ubiquitous Information Technologies and Applications, Vol. 280, Springer, Berlin, Heidelberg, 2014, pp. 409–416 (Chapter).

[18] Y. Kwong Kwok, I. Ahmad, Static scheduling algorithms for allocating directed task graphs to multiprocessors, ACM Comput. Surv. 31 (4) (1999) 406–471.

[19] Y.C. Lee, R. Subrata, A.Y. Zomaya, On the performance of a dual-objective optimization model for workflow applications on grid platforms, IEEE Trans. Parallel Distrib. Syst. 20 (9) (2009) 1273–1284.

[20] Y.C. Lee, A.Y. Zomaya, On effective slack reclamation in task scheduling for energy reduction, J. Inf. Process. Syst. 5 (4) (2009) 175–186.

[21] M. Levin, Storage management disciplines are declining, June 2006. URL: http://www.computereconomics.com/article.cfm?id=1129.

[22] J. Li, S. Su, X. Cheng, Q. Huang, Z. Zhang, Cost-conscious scheduling for large graph processing in the cloud, in: Proc. of the 13th International Conference on High Performance Computing and Communications, HPCC, 2011, pp. 808–813.

[23] X. Lin, C. Wu, On scientific workflow scheduling in clouds under budget constraint, in: Proc. of the 42nd International Conference on Parallel Processing, ICPP, 2013, pp. 90–99.

[24] J. Livny, H. Teonadi, M. Livny, M. Waldor, High-throughput, kingdom-wide prediction and annotation of bacterial non-coding RNAs, PLoS ONE 3 (9) (2008) e3197.

[25] V. Lynch, J. Cobb, E. Farhi, S. Miller, M. Taylor, Virtual experiments on the neutron science teragrid gateway, in: TeraGrid, 2008.

[26] J.C. Maces, A. van Moorsel, P. Watson, The case for dynamic security solutions in public cloud workflow deployments, in: Proc. of IEEE/IFIP 41st International Conference on Dependable Systems and Networks Workshops, DSN-W, 2011, pp. 111–116.

[27] M. Malawski, G. Juve, E. Deelman, J. Nabrzyski, Cost- and deadline-constrained provisioning for scientific workflow ensembles in IaaS clouds, in: Proc. of the International Conference on High Performance Computing, Networking, Storage and Analysis, SC, 2012, pp. 1–11.

[28] M. Mao, M. Humphrey, Scaling and scheduling to maximize application performance within budget constraints in cloud workflows, in: Proc. of IEEE 27th International Symposium on Parallel & Distributed Processing, IPDPS, 2013, pp. 67–78.

[29] P. Mell, T. Grance, The NIST definition of cloud computing, version 15, October 2009. URL: http://csrc.nist.gov/groups/SNS/cloud-computing.

[30] A. Oprescu, T. Kielmann, Bag-of-tasks scheduling under budget constraints, in: Proc. IEEE Second Int. Cloud Computing Technology and Science Conf., 2010, pp. 351–359.

[31] R. Prodan, M. Wieczorek, Bi-criteria scheduling of scientific grid workflows, IEEE Trans. Autom. Sci. Eng. 7 (2) (2010) 364–376.

[32] A. Quiroz, H. Kim, M. Parashar, N. Gnanasambandam, N. Sharma, Towards autonomic workload provisioning for enterprise grids and clouds, in: Proc. 10th IEEE/ACM Int. Grid Computing Conf., 2009, pp. 50–57.

[33] L. Ramakrishnan, B. Plale, A multi-dimensional classification model for workflow characteristics, in: Workflow Approaches to New Data-Centric Science, with ACM SIGMOD, 2010, pp. 1–12.

[34] M. Singhal, S. Chandrasekhar, T. Ge, R. Sandhu, R. Krishnan, G.-J. Ahn, E. Bertino, Collaboration in multicloud computing environments: framework and security issues, IEEE Comput. 46 (2) (2013) 76–84.

[35] S. Song, K. Hwang, Y.-K. Kwok, Risk-resilient heuristics and genetic algorithms for security-assured grid job scheduling, IEEE Trans. Comput. 55 (2006) 703–719.

[36] S.X. Sun, Q. Zeng, H. Wang, Process-mining-based workflow model fragmentation for distributed execution, IEEE Trans. Syst. Man. Cybern. Part A 41 (2) (2011) 294–310.

[37] X. Tang, K. Li, Z. Zeng, B. Veeravalli, A novel security-driven scheduling algorithm for precedence constrained tasks in heterogeneous distributed systems, IEEE Trans. Comput. 60 (7) (2011) 1017–1029.

[38] J.L. Tilson, G. Rendon, M.-F. Ger, E. Jakobsson, MotifNetwork: A grid-enabled workflow for high-throughput domain analysis of biological sequences: Implications for annotation and study of phylogeny, protein interactions, and intraspecies variation, in: Proc. 7th IEEE Int. Conf. Bioinformatics and Bioengineering BIBE 2007, 2007, pp. 620–627.

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

[40] Y. Wang, P. Lu, Dataflow detection and applications to workflow scheduling, Concurr. Comput.: Pract. Exp. 23 (2011) 1261–1283.

[41] Y. Wang, W. Shi, On scheduling algorithms for MapReduce jobs in heterogeneous clouds with budget constraints, in: Principles of Distributed Systems, Vol. 8304, Springer International Publishing, 2013, pp. 251–265 (Chapter).

[42] P. Watson, A multi-level security model for partitioning workflows over federated clouds, in: Proc. of IEEE Third International Conference on Cloud Computing Technology and Science, CloudCom, 2011, pp. 180–188.

[43] T. Xie, X. Qin, Performance evaluation of a new scheduling algorithm for distributed systems with security heterogeneity, J. Parallel Distrib. Comput. 67 (2007) 1067–1081.

[44] J. Yu, R. Buyya, A budget constrained scheduling of workflow applications on utility grids using genetic algorithms, in: Proc. Workshop Workflows in Support of Large-Scale Science, 2006, pp. 1–10.

[45] J. Yu, R. Buyya, C.K. Tham, Cost-based scheduling of scientific workflow applications on utility grids, in: Proc. of First International Conference on e-Science and Grid Computing, 2005. http://dx.doi.org/10.1109/E-SCIENCE.2005.26.

[46] D. Yuan, Y. Yang, X. Liu, J. Chen, A data placement strategy in scientific cloud workflows, Future Gener. Comput. Syst. 26 (8) (2010) 1200–1214.

[47] D. Yuan, Y. Yang, X. Liu, J. Chen, On-demand minimum cost benchmarking for intermediate dataset storage in scientific cloud workflow systems, J. Parallel Distrib. Comput. 71 (2) (2011) 316–332.

[48] L. Zeng, B. Veeravalli, X. Li, ScaleStar: budget conscious scheduling precedence-constrained many-task workflow applications in cloud, in: Proc. of the 26th International Conference on Advanced Information Networking and Applications, AINA, Fukuoka, Japan, 2012, pp. 534–541.

[49] H. Zhao, R. Sakellariou, Scheduling multiple DAGs onto heterogeneous systems, in: Proc. 20th International Parallel and Distributed Processing Symposium, IPDPS, 2006.

[50] W. Zheng, R. Sakellariou, Budget-deadline constrained workflow planning for admission control, J. Grid Comput. 11 (4) (2013) 633–651.

[51] Q. Zhu, G. Agrawal, Resource provisioning with budget constraints for adaptive applications in cloud environments, in: Proc. of the 19th ACM International Symposium on High Performance Distributed Computing, HPDC'10, 2010, pp. 304–307.

Lingfang Zeng received his B.S. in Applied Computer from Huazhong University of Science and Technology (HUST), Wuhan, China in 2000, M.S. in Applied Computer from China University of Geoscience, China in 2003 and Ph.D. in Computer Architecture, from HUST in 2006. He was Research Fellow for over four years in the Department of Electrical and Computer Engineering, National University of Singapore, Singapore, during 2007–2008 and 2010–2013. He is currently with Wuhan National Lab for Optoelectronics, and School of Computer, HUST, as an associate professor. He published more than 40 papers in major journals and conferences. He is a member of IEEE.

Bharadwaj Veeravalli, Senior Member, IEEE & IEEE-CS, received his B.Sc. in Physics, from MDU-Kam University, India (1987), Master's in Electrical Communication Engineering from Indian Institute of Science, Bangalore, India (1991) and Ph.D. from the Department of Aerospace Engineering, Indian Institute of Science, Bangalore, India (1994). He was a post-doctoral fellow in the Department of Computer Science, Concordia University, Montreal, Canada (1994–1996). He is currently a tenured Associate Professor with the Department of Electrical and Computer Engineering, in NUS, Singapore. He is currently serving the Editorial Board of IEEE Transactions on Computers, IEEE Transactions on SMC-A, etc., as an Associate Editor.

Xiaorong Li received Ph.D. from Electrical and Computer Engineering Department at the National University of Singapore in 2006 and B.E. from Beijing University of Posts and Telecommunication, China in 1998. She is a research scientist and currently the manager of distributed computing capability group in the A*STAR Institute of High Performance Computing, Singapore. Her research interests include distributed computing systems, data analytic, and QoS management in Cloud/Grid Computing. She is a member of IEEE and ACM.