Cost Effective, Reliable, and Secure Workflow Deployment over Federated Clouds

Cost Effective, Reliable and Secure Workflow Deployment over Federated Clouds

Zhenyu Wen*, Jacek Cała†, Paul Watson‡ and Alexander Romanovsky§
School of Computing Science, Newcastle University
*Email: *z.wen, †jacek.cala, ‡paul.watson, §alexander.romanovsky@newcastle.ac.uk*

***Abstract*—The federation of clouds can provide benefits for cloud-based applications. Different clouds have different advantages – one might be more reliable whilst another might be more secure or less expensive. However, being able to select the best combination of clouds to meet the application requirements is not trivial.**

This paper presents a novel algorithm to deploy workflow applications on federated clouds. Firstly, we introduce an entropy-based method to quantify the most reliable workflow deployments. Secondly, we apply an extension of the Bell-LaPadula Multi-Level security model to meet application security requirements. Finally, we optimise deployment in terms of its entropy and also its monetary cost, taking into account the price of computing power, data storage and inter-cloud communication.

To evaluate the new algorithm we compared it against two existing scheduling algorithms: Dynamic Constraint Algorithm (DCA) and Biobjective dynamic level scheduling (BDLS). We show that our algorithm can find deployments that are of equivalent reliability but are less expensive and also meet security requirements. We have validated our solution using workflows implemented in the e-Science Central cloud-based data analysis system.

***Keywords*-Cloud Computing; Reliability; Security; Workflow; Scheduling; Cost**

I. Introduction

Cloud computing has experienced significant growth over recent years with a multitude of public clouds becoming available. There has been also a trend toward managing local data centres as private clouds. Thus, currently application developers who decide to host their system in the cloud face the issue of which cloud to choose to best meet their requirements in terms of price, reliability and security. The decision becomes even more complicated if the application consists of a number of distributed components, each with slightly different requirements.

Access to a range of different cloud providers has led to an interest in the idea of federated clouds [1] and [2]. Cloud federation gives the ability to distribute a single application on two or more cloud platforms, so that the application can benefit from the advantages of each of them. This creates the problem of how to find the best distribution (or deployment) of application components, that yields the most gains. In this paper we tackle this problem and propose an algorithm to deploy workflow-based applications over federated clouds in order to exploit the strengths of each cloud in terms of reliability, security and price.

Our algorithm schedules an application structured as a workflow so as to reliability and security requirements while minimising the financial cost of execution.

The algorithm extends our security model presented previously in [3], and adapts it to the new, multi-criteria requirements. It is based on the Bell-LaPadula Multi-Level Security model and was designed to deploy a workflow over a federated cloud to meet certain security requirements.

The basis for the algorithm is a new method to quantify the most reliable workflow deployments which applies Shannon's information theory [4]. Using reliability information for each application component and the underlying cloud platform, the method calculates the entropy of a workflow deployment. This value is then used as a constraint in the optimisation problem. We argue that by using entropy we can reduce the overall risk of workflow failures caused by a small number of components being deployed on less reliable clouds.

Furthermore, as there may be a number of deployments that meet our security and reliability requirements, we search for the option that minimises the monetary cost. Finding the optimal deployment is, however, an NP-hard problem, thus we need an approximate algorithm to solve it. The two most common approaches are: (1) to linearise the problem by assigning weights to the criteria and then optimise the weighted sum, (2) to optimise one criterion and keep the others constrained within predefined thresholds. In the first approach the difficulty is not only in defining the weights properly but also in the limitation of the simple, linear model which may not be able to accurately represent the complexity of the problem. Hence in this work we use the second approach.

To handle this multi-criteria and NP-hard problem we generate a valid initial solution and then apply a set of refinement methods to approach the optimum. At the same time we want guarantees that the time complexity of the algorithm is polynomial.

In summary the contributions of this paper are as follows:

- An algorithm that reduces the time taken to derive a deployment of a workflow application across federated clouds is presented. The algorithm takes into account security and reliability requirements and reduces the monetary cost incurred from three main sources in the cloud: computation, data transfer and data storage.
- A novel method to quantify the reliability of a workflow

deployment using entropy, remedying the limitation of the existing methods.
- An evaluation of our work using both randomly generated workflows and an existing scientific workflow running on e-Science Central, a cloud-based data analysis platform.

The remainder of the paper is structured as follows. We first introduce the notation and models used to represent the reliability and security requirements and to calculate the monetary cost of deployment. Next, in Section III we show the optimisation problem as described by the models. Then a scheduling algorithm to search efficiently for a suitable deployment option is presented. In Section V we discuss the evaluation of our work. Finally, future work is outlined and conclusions are drawn.

II. Motivation and Problem Description

With the increasing availability of public and private cloud resources it is easy to deploy instances of the same service in multiple places. We observe this tendency with our e-Science Central data analysis system [5] which, depending on the use case, has been deployed in a variety of locations including private clouds at universities in Spain and Brazil and public cloud resources such as Amazon AWS and Microsoft Azure. Each of these clouds has its own advantages and thus we focus in this paper on how a single workflow application might be deployed over the federated cloud. By federated cloud we consider in this paper a set of workflow execution environments (such a e-Science Central) running in different clouds. Our goal is to partition a workflow application in such a way that it can benefit from the "best" combination of these environments.

In this section we introduce the notation used and define the three concepts that form the basis for our algorithm: the measure of reliability, the security rules and the cost model.

A. Reliability

The target clouds of this paper are comprised of a set of PMs (physical machines); the set of clouds can be geographically distributed. A single physical machine pm_i is able to contain $n \in [0, 1, ...]$ VMs (virtual machines). Each VM can run 0 or more instances of WPs (workflow execution platform). Lastly, a WP can run a number of services s_i concurrently.

Clearly, for a service to be executed completely and reliably on a workflow execution platform, all elements of this vertical stack (PM, VM, WP and service) must run fault-free during service execution.

1) Computing Reliability: reliability (REL) defined as: $REL = \frac{\text{Fault-Free Time}}{\text{Total Time}}$. Let $\hbar_{pm}$ be a random variable which represents the time of the failure of machine pm while f_{pm} denotes the probability density function of $\hbar_{pm}$. We assume that failures are randomly distributed in time and can be modelled by the exponential distribution. Therefore, an exponential probability density function is $f_{pm}(t) = \lambda_{pm} e^{-\lambda_{pm} t}$, where λ_{pm} is the failure rate of machine pm. Consequently, the reliability function of pm can be defined as in [6]:

$$R_{pm}(t) = 1 - \int_0^t f_{pm}(t)dt = e^{-\lambda_{pm} t} \tag{1}$$

Similarly, the reliability of VM and the workflow execution platform can be denoted by $R_{vm}(t) = e^{-\lambda_{vm} t}$ and $R_{wp}(t) = e^{-\lambda_{wp} t}$ respectively.

2) Measure of Workflow Reliability: We assume workflow $\mathcal{W}$ consists of n services $s_1, \ldots, s_n$, and the reliability of each service is R_{s_i}. Entropy [4] is a widely used measurement that captures the degree of dispersal or concentration of random variable distributions. For a discrete random variable X with the probabilities $p(x_i)$ it is defined as:

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log p(x_i) \tag{2}$$

We deem $p(x_i)$ as the reliability of s_i. Therefore the entropy of the workflow $\mathcal{W}$ is:

$$H(\mathcal{W}) = -\sum_{i=1}^{n} R_{s_i} \log R_{s_i} \tag{3}$$

Why Entropy? The most widely used methods to measure the reliability of a system are arithmetic mean ($\frac{\sum_{i=1}^{n} REL_i}{n}$) and power ($\prod_{i=1}^{n} REL_i$). For workflow-based applications the most popular method is power [7],[8] and [9]. However, we decided to use another way to quantify reliability because both the mean and power method share the same limitation – they allow mixing many high reliable services with a few services of lower reliability.

For example, let us assume that the required reliability rate of workflow $\mathcal{W}$ is $REL = 0.85$, and that it consists of 3 services s_1, s_2 and s_3. Both distributions: $DT_1 = (0.97, 0.985, 0.89)$ and $DT_2 = (0.95, 0.95, 0.95)$ meet the required threshold. Yet, if $\mathcal{W}$ is a pipeline structured as $s_1 \rightarrow s_2 \rightarrow s_3$, the last service for DT_1 has relatively high chances to fail which increases the risk of the loss of work done by s_1 and s_2.

The method based on entropy has more advantage of reflecting a situation when a fault in one element limits the result and can not be compensated by other elements.

It because Entropy is a version of weighted geometric mean, and power belong to geometric mean. In our case weighted geometric mean can include more information about service reliability than geometric mean. Therefore, *weighted geometric mean better reflects a situation when a shortage in one element limits the result and cannot be compensated by other elements. At the same time, keep the characteristic of power which guarantees the reliability constraint*

B. Security Rules

A security model is needed to determine whether a deployment of services and data onto a set of clouds meets organisation's security requirements. We use the security model defined in our previous work which builds upon the Bell-LaPadula model.

In the following we will use $\lambda \in \Lambda$ to denote a secure distribution of a workflow deployed on a set of clouds; where Λ is all possible distributions of the workflow.

C. Cost Model

The cost model plays key role in our algorithm. We assume that clouds are linked in a fully connected topology and data can be freely transferred between them. Additionally, a wp can run several services at the same time.

To calculate the cost of executing a service in the federated cloud we define a set of cost functions. First is the data storage cost:

$$SCOST(s_i^p) = \sum_{d_{i,j} \in OUT} d_{i,j} \times T_{i,j} \times Store_p \quad (4)$$

Where s_i^p means service s_i is deployed on workflow/cloud platform p. OUT is a set of data dependencies, representing that data are generated by s_i and transferred to its immediate successor s_j which is not deployed on platform p (note that if all immediate successors of s_i are on p, $OUT = \emptyset$). $d_{i,j}$ represents the amount of data which is generated by s_i and consumed by s_j. $T_{i,j}$ denotes storage time of data $d_{i,j}$, which is the time from the data being generated until the end of workflow execution. Finally, $Store_p$ is the cost of storing 1GB of data for 1 hour on workflow platform p.

Only a few researchers consider the data storage cost when deploying workflows over a federated clouds. However, the following four reasons describe why it is worth taking into account: 1) it makes the cost model more complete because cloud providers usually charge not only for compute resources and data transfer but also for data storage; 2) for data intensive workflows the storage cost becomes considerable, especially when data need to be transmitted between two clouds; 3) storing the output of a workflow partition is a checkpoint which can reduce the loss in the event of an outage in the clouds running the following partitions; 4) such data checkpointing mechanisms are implemented by some workflow management platforms. For example, the e-Science Central cloud platform [5] implicitly stores the data that need to be transferred between partitions.

Next function, $CCOST$, is used to estimate the communication cost for services:

$$CCOST(s_j^q) = \sum_{d_{i,j} \in IN} d_{i,j} \times Com_{p,q} \quad (5)$$

It is the cost of the data transferred from the immediate predecessors of service s_j (denoted as IN). $Com_{p,q}$ represents the unit cost of transferring 1GB of data from workflow platform p to q. Note that if two services are deployed on the same platform, the unit cost is zero, i.e. $\forall p = q : Com_{p,q} = 0$.

Finally, $ECOST(s_j^p)$ indicates the execution cost of service s_j on platform p. It is defined as:

$$ECOST(s_j^p) = T_j^p \times Exec_p \quad (6)$$

Where T_j^p is the execution time of s_j on platform p, and $Exec_p$ represents the cost of using compute resources on p for 1 hour.

Based on these three functions, we can define the total cost of deploying a workflow over a set of clouds:

$$COST(\lambda') = \sum_{s_i^p \in \mathcal{W}} SCOST(s_i^p) + CCOST(s_i^p) + ECOST(s_i^p) \quad (7)$$

Where λ' is one of the deployment solutions, $\lambda' \in \Lambda$.

III. Taxonomy of Scheduling Criteria

A. Multi-Objective Optimisation Problem

For a given λ, a secure deployment of workflow $\mathcal{W}$ over federated clouds P, we propose to optimise two parameters: i) minimise the value of entropy $H(\lambda)$ which results in a distribution that maximises the reliability of the workflow, ii) minimise the value of $COST(\lambda)$ to obtain deployments with low cost of execution. We express this problem as:

$$\textbf{min}\,(COST(\lambda))$$
$$\textbf{s.t.:}\, \exists\, \text{secure deployment}\, \lambda \;\&\&\; \textbf{min}\,((H(\lambda))$$

Finding a solution which optimises both $COST(\lambda)$ and $H(\lambda)$ is a challenging problem and the main focus of our algorithm.

IV. Scheduling Algorithm

We propose a novel scheduling algorithm that can optimise the deployment of a workflow over a set of clouds. It takes into account user requirements against multiple criteria, namely security, reliability and cost. The optimisation part of the algorithm is an extension of the multiple-choice knapsack problem (MCKP) [10].

Overall, our algorithm is executed following three steps: 1) set a boundary on one of the two objectives, 2) search for a deployment which minimises the other objective while the first objective is within the boundary chosen and 3) traverse the available options to optimise the deployment found in step 2.

In the first step we set a bound $\mathcal{C}$ on entropy such that:

$$H(\lambda) \leq \frac{H(\lambda^C) + H(\lambda^E)}{2} = \mathcal{C} \quad (8)$$

where λ^E is the entropy-optimal deployment, and λ^C is the cost-optimal deployment. We could also make a bound on cost, however, we need to guarantee that the reliability rate is acceptable. We also do not set the boundary on security because all candidate solutions we generated meet the security requirements W.

As mentioned above, our optimisation is an extension of MCKP. MCKP is used to optimise a set of decisions $S = \{s_1, \ldots, s_n\}$ within a defined constraint. In our scenario, S represents the set of deployments Λ, where s_i is a mapping of each workflow object o onto a cloud platform p, $s_i = \{o_1^{p_x}, \ldots, o_m^{p_z}\}$. Our algorithm aims to find the optimal s_i.

To realise this goal, we firstly generate the cost- optimal and entropy-optimal deployments that satisfy security requirements. If the cost-optimal solution meets the reliability requirement, then this will be the optimal option. However, in most cases the cost-optimal solution does not meet the reliability constraints invoking a set of optimisation mechanisms to improve this deployment. Thus, our algorithm consists of two phases: initial deployment and deployment optimisation.

A. Initial Deployment

The goal of the initial deployment is to generate a solution which can be a seed to find the optimal deployment. To generate the cost-optimal deployment (λ^C) we applied the NCF algorithm from our previous work [11]. The algorithm works as follows: 1) A greedy-based function determines a deployment which accounts for only the local costs related to each single service considered in isolation. 2) Based on the first step we have more information that can be used to produce the optimal deployment and to make a short term sacrifice for long term benefit. Obviously, for each step the security requirements must be guaranteed. Then, we compare the entropy value of λ^C with constraint $\mathcal{C}$. If $H(\lambda^C) < \mathcal{C}$, the λ^C will be the optimal solution. In contrast, we use an entropy-optimal solution which can be generated by applying a basic greedy algorithm as a seed to find the optimal solution. The reason why the entropy-optimal solution can be generated by the basic greedy algorithm is because the entropy values of each service are independent of each other. Thus, the problem has the optimal substructure.

B. Deployment Optimisation

If λ^C is not a valid deployment, we use λ^E as the seed to find available options. We limit the search to find t deployment options and store the result in descending order of the cost. The optimal solution is the last on the list; the pseudocode is shown in Algorithm 1.

Finding t valid options was the key challenge in designing the algorithm because of the huge search space. To solve it we observe the contribution of each cloud for each service, also taking into account its predecessors. To calculate the cost of deploying a service on a specific cloud we use the COD function. COD is calculated by adding the computing cost of service s_i to the transmission cost and storage cost of data sent from all of its immediate predecessor services that are not in the same cloud.

Algorithm 1 Deployment Optimisation

$\mathcal{W}$ – workflow; λ^E – the entropy-optimal deployment; $\mathcal{C}$ – entropy constraint; t – maximum number of deployments; $Cloud$ – valid clouds for each service; M – valid deployments;
$M[0] \leftarrow \lambda^E$
$\lambda \leftarrow \lambda^E$
for $i \in 1...t$ **do**
  $\lambda^{NEW} \leftarrow CHANGE(\lambda, Cloud)$; ▷ generate a new deployment from λ
  $\lambda \leftarrow \lambda^{NEW}$
  $M[i] \leftarrow \lambda^{NEW}$
end for
sort M by cost **return** $M[t]$

$$COD(s_i^p) = SCOST(s_i^p) + CCOST(s_i^p) + ECOST(s_i^p)$$

Each s_i may have more than one valid cloud, therefore the ranking of the valid clouds can be created by the ascending order of the COD value. This is described by the ranking algorithm (Algorithm 2).

Algorithm 2 Rank

function RANK(λ)
  ▷ Topsort returns a topological order of $\mathcal{W}$
  for $s_i \in topsort(\mathcal{W})$ **do**
    for $p \in Cloud[s_i]$ **do**
      $Tmp[p] \leftarrow COD(s_i^p)$
    end for
    sort Tmp in the ascending order
    ▷ update the order of the clouds in $Cloud[s_i]$ by following the corresponding order of cache
    $Cloud[s_i] \leftarrow Tmp$
  end for
end function

As Algorithm 3 shows, the ranking is used to find an alternative cloud for s_i, which is chosen randomly by using the *Benford* function.

The reason we used randomisation is because the optimal option is not always composed of the clouds which minimise the COD. By applying the function, the clouds on the front of the ranking have a higher chance of being selected but we also avoid the situation that some clouds are never chosen.

Algorithm 3 Change

function CHANGE($\lambda, Cloud$)
▷ Sort the cloud list by COD in the ascending order and update the Cloud list for the Benford function
$Cloud \leftarrow RANK(\lambda)$
while true **do**
Randomly choose a service s_i
$\lambda^{NEW} \Leftarrow Cloud[Benford(s_i)]$; ▷ Apply $Benford$ function to select a replace the cloud for s_i
if $\lambda^{NEW} \notin M$ and is secure and $H(\lambda^{NEW}) \leq \mathcal{C}$ **then return** λ^{NEW}
end if
end while
end function

The *Benford* function is a transformation of Benford's law [12] defined as:

$$Benford(s_i) = \lfloor \frac{1}{10^b - 1} \rfloor$$
$$b = random(\log_{10}(1 + \frac{1}{len(Cloud[s_i])}), log_{10}2)$$

V. Results of the Experiments and Evaluation

In order to evaluate our algorithm we set up two experiments. The first used randomly generated workflows to compare our algorithm with other existing algorithms. In the second experiment we applied our algorithm over federated clouds to deploy a scientific workflow from one of the projects we have been involved in. Note that in this work, workflows are defined as directed acyclic graphs where vertices represent tasks and arcs between them represent data dependencies.

A. Randomly Generated Workflows

To define a workflow we use distance matrix $D = [d_{i,j}]$. Each $d_{i,j}$ greater than 0 means that service s_i is an immediate predecessor of s_j and randomly assigned value $d_{i,j}$ represents the amount of data transferred between the services.

In this experiment, we consider five different clouds that are assigned the start time and failure rate randomly in predefined ranges. For example, the clouds may be assigned start times as 20–50 hours before deploying a workflow application. The initial failure rate of each cloud may be between 0.3–0.5%. Therefore, if a cloud has been executed 20 hours before deploying a workflow and its failure rate is 0.3%, its entropy value will be $e^{-0.003\times20} = 0.941$. The other parameters such as cloud security levels, service security constraints and service execution times are also generated randomly.

Furthermore, as discussed in section IV-A, we can easily get the entropy values of both cost-optimal deployment λ^C and entropy-optimal deployment λ^E, therefore setting the constraint of the entropy as $C = \frac{H(\lambda^C)+H(\lambda^E)}{2}$.

The experiments are implemented in Java and run on a 4-core machine with 2 GHz Intel i7 processor, 8G RAM and OS X Yosemite.

We compare our algorithm EMCK (extended multiple-choice knapsack) with DCA [13] and BDLS [8] which represent two existing and widely used multi-criteria scheduling algorithms. DCA and our algorithm both extend MCKP. However, DCA is focused on a single computing resource, and does not take into account cost of data transfer and storage. BDLS is a list scheduling algorithm that schedules services according to a priority list of service–resource pairs. For the purpose of the experiment, we had to extend and adapt BDLS as follows: (1) we applied the ranking mechanism, shown in Algorithm 2, to build cost list CL; (2) we added another list with entropy constraint for each service defined as $EL_{s_i} = \frac{H(s_i^C)+H(s_i^E)}{2}$, where $H(s_i^C)$ represents the entropy value of s_i in the cost-optimal deployment, whereas $H(s_i^E)$ is the value in the entropy-optimal deployment. As a result the services were assigned to the cloud which minimised the cost and also met the entropy requirement.

The relative monetary cost of all algorithms is displayed in Figures 1 and 2. In all tested cases our EMCK algorithm gave cheaper deployments than the other two. Comparing with the entropy-optimal solution, EMCK significantly reduced the monetary costs while keeping the reliability within the constraints.

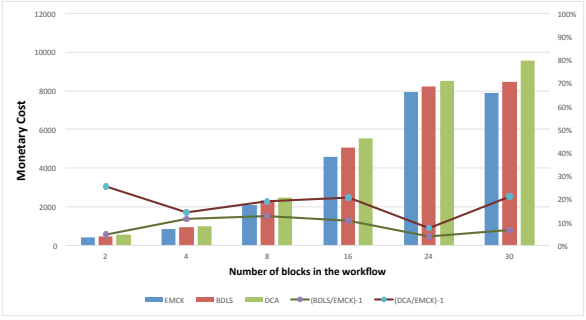

Figure 1: The Cost Comparison with Other Algorithms.

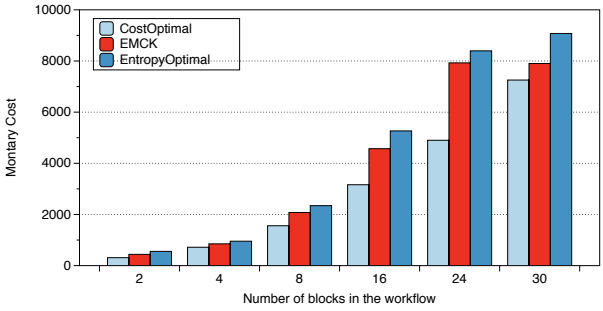

Figure 2: The Cost Comparison with Optimal Deployments.

Figure 3 presents the execution time of EMCK with all results below 1 second. There are significant fluctuations in the execution time because of a few cases when the cost-optimal deployment was also the entropy-optimal deployment, and thus it took less than 100 μs to find them. The comparison between the execution time of EMCK and other algorithms is presented in Figure 4. Due to large differences between the algorithms we present in the figure the ratio $EMCK/$other algorithm. Despite our algorithm was much slower than BDLS, still the execution time was acceptable (below 1 second). It is a polynomial time algorithm and its response time is perfectly sufficient to handle our use case.

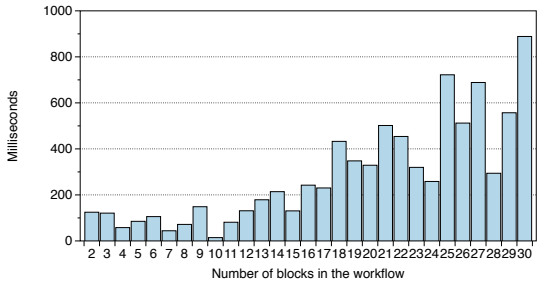

Figure 3: Execution time of finding an optimal deployment.

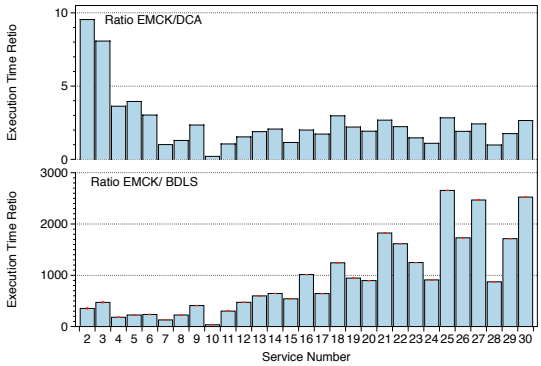

Figure 4: Ratio between execution times with the other two algorithms.

B. Scheduling Scientific Workflows

To evaluate our algorithm in conditions closer to a production use we applied it to schedule scientific workflows in e-Science Central [5]. e-Science Central (e-SC) is a cloud-based data analysis system that can run on a range of public and private clouds including Amazon AWS, Microsoft Azure and OpenShift. e-SC is a SaaS and PaaS, it offers a web user interface but also provides a range of APIs to allow users to control the system from code. For example, the storage subsystem API allows users to upload, download and manipulate data, whereas the workflow API enables them to execute, terminate and monitor workflow invocations. We used the APIs to create a tool that can orchestrate invocations

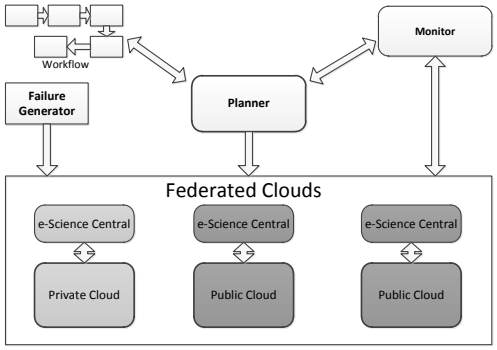

Figure 5: The architecture of the deployment tool.

of a single workflow partitioned over a number of e-SC instances.

1) Tool Design: Figure 5 shows the architecture of our deployment tool. It consists of four core components: *Planner* assigns workflow partitions to *Federated Clouds* using the algorithm discussed above. The *Federated Clouds* is a set of e-Science Central instances which run in different clouds and are accessed by other components via e-SC APIs. *Monitor* observes the status of each instance, detects failures, and provides the information about available instances to the *Planner*. Finally, *Failure Generator* is used to simulate failures by shutting down e-SC instances with predefined probability.

2) Experiment Setup: To verify our algorithm we selected one of the workflows used in the Cloud e-Genome project [14] (see Figure 6). The project implements a whole exome sequencing pipeline using e-Science Central workflows deployed on the Microsoft Azure cloud.

Whilst in the project the security aspects are not of primary concern, guaranteeing that human genomic data can be securely processed on the cloud is very important. Therefore, we modelled the security requirements of a selected Cloud e-Genome workflow by assigning security levels as shown in Tables I and II; note that the size of data transferred between blocks and the execution time of each block are actual values taken from logs collected by e-SC. Table I shows data sizes in GB, where 0 denotes less than 1 MB of data. The pricing shown in Table III is collected from two major cloud providers and is based on the equivalent VM configurations, referring to public and private cloud.

To simulate this environment we set up three virtual machines each running a single instance of the e-SC system. VM1 was hosted on a personal PC and represented the private cloud. Two other VMs were hosted in our University virtualised environment and played the role of public cloud providers Pu1 and Pu2.

In order to test our algorithm the platform's start-up time

Service Name	Id	Clearance	Location	Time [h]
Sample Name	S_1	1	0	1
Import Input File	S_2	1	0	1.5
String List	S_3	1	0	3
Prepare HG19	S_4	1	0	0.1
GATK Filter	S_5	2	0	10
Import Exome-Regions	S_6	1	0	7
Interval padding	S_7	0	0	20
Column Join	S_8	2	0	0.1
Annotate Sample	S_9	2	0	5
Export CSV	S_{10}	1	0	0.3

Table I: Service security requirements and execution times.

Data	Location	Size (GB)
$S_{1,8}$	1	0
$S_{2,5}$	0	1.1
$S_{3,8}$	2	0.01
$S_{4,5}$	0	0.005
$S_{4,7}$	0	0.005
$S_{5,7}$	0	6.2
$S_{6,7}$	0	10.3
$S_{7,9}$	1	3.6
$S_{8,9}$	0	0
$S_{9,10}$	0	0.05

Table II: Data security requirements.

Cloud	Pr1	Pu1	Pu2
Security	2	1	0
CPU	3.41(/h)	2.40(/h)	1.28(/h)
Pr1	0	0.1	0.11
Pu1	0.13	0	0.09
Pu2	0.07	0.02	0

Table III: Basic attributes of the three clouds used in the experiment: security level, cost of computing resources, cost of data transfer between clouds (e.g. $Pr1 \rightarrow Pu1 = 0.1$).

Service	Cheapest	EMCK-optimal
S_1	Pr1	Pu1
S_2	Pu2	Pu2
S_3	Pr1	Pr1
S_4	Pu2	Pu2
S_5	Pu2	Pu1
S_6	Pu2	Pu2
S_7	Pu2	Pu2
S_8	Pr1	Pr1
S_9	Pu2	Pu2
S_{10}	Pu2	Pr1

Table IV: Two deployments.

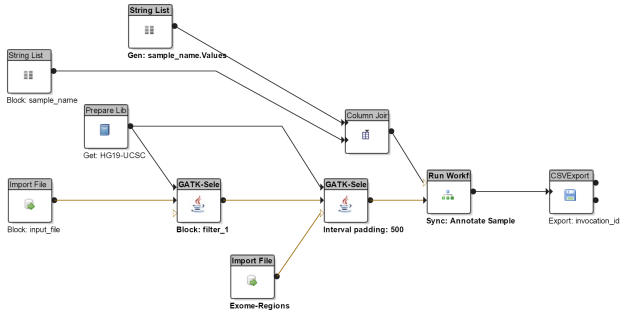

Figure 6: A selected workflow from the Cloud e-Genome project.

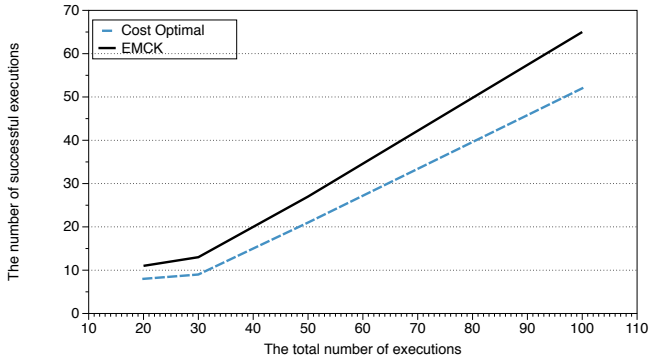

Figure 7: The execution results of two deployments.

must be defined. It is the time when a platform was started and it is needed to calculate the platform reliability value at the moment when a workflow deployment occurs. We set the failure rate of each VM as 0.03, reference start-up time of VM1, VM2 and VM3 as 1.7h, 2.4h and 3.5h, and their initial reliability as 0.95, 0.93 and 0.90, respectively. The reason for the high failure rates is that we need failures to occur while we are running the experiments. Otherwise, we would need to run the experiments for a prohibitively long time. For the same reason we reduced the execution time of each workflow to about 30 seconds, not as shown in Table I, by scaling down the amount of input data by a factor of 2400.

3) Results and Discussion: Based on the presented experiment setup our algorithm generated two deployments (Table IV): the cheapest and the EMCK-optimal with costs 69.832 and 128.897 respectively. Figure 7 shows that the deployment produced by our algorithm is more reliable than the cheapest one by about 25%.

Someone may challenge that the failure rate for the EMCK-optimal deployment is too high (about 40%). This, however, is the result of the high initial failure rate set for each VM.

VI. Related Work

Workflow scheduling has been a classic research topic for decades and has developed together with changes in the technology. In the last decade, most work has focused on workflow mapping problems using DAG scheduling heuristics such as [15], to name just a few. However, these algorithms are all based on single computing resources such as "free" grid resources, and thus aim to minimise makespan. Furthermore, there are a few works which consider other scheduling with other objectives such as security, reliability and performance. Song introduces in [16] a version of a genetic algorithm to assign jobs based on risk-resilient strategies to provide security assurance of trusted Grid computing. The algorithm in [17] tolerates processor failure by means of primary/backup techniques to allocate two copies of each task to different processors.

In [18], the authors model the security needs of the real-time application on clusters, and design and implement a

scheduling algorithm, including the security-aware heuristic strategy. In contrast, in our work we consider not only one objective or criteria but we designed an algorithm which can optimise security, reliability and monetary costs together.

Multi-objective optimisation has been explored previously in heterogeneous computing. The work in [8] presents two algorithms to address the trade-off between makespan and reliability. One is based on a dynamic level scheduling algorithm and the other is a version of a genetic algorithm. However, our work uses a model-based technique on federated clouds where the monetary cost plays a crucial role, and none of the mentioned algorithms consider this.

Research related to federated clouds or multi-cloud environments is still new with little literature available and most focus is merely on a single objective. For example, in [19], the authors introduced a pricing model and truthful mechanism for scheduling workflow applications to different clouds. However, reliability and security are not considered.

Finally, Web Services has been developed for about two decades and resulted in a number of service selection algorithms that are related to our work [20], [21]. However, service selection and discovery techniques are focused on grouping business processes and the existing services to create new applications. In our work we tackle the opposite problem which is to deploy and schedule an existing scientific workflow applications. Scientific workflows often require large amounts of data to be transferred and thus factors such as the cost of data transfer between two clouds require a new approach. Similarly, the security model used in our work makes it different from other work in this area.

VII. Conclusion

Cloud computing provides elasticity for deploying services or cloud based platforms over different clouds, this SOA fashion makes federated clouds become possible. Furthermore, the pay-as-you-go utility model means users can easily calculate the monetary cost of computing resources that have been used. In this paper, we design and implement an algorithm to solve the problem of deploying workflow applications over federated clouds meeting the reliability, security and monetary requirements. We also develop a tool (included the scheduling algorithm) to allocate a workflow over a set of e-Science Central instances which are running on different clouds. We have shown the trade-off between reliability and cost when the location value is set as hard bounded. Our algorithm guarantees the reliability and security constraint and optimises the cost. The evaluation has been done by using randomly generated workflow and a real world cloud based platform. Experimental results prove that our solution can guarantee the security and reliability to find a cost-optimal deployment.

Future work will consider adding more evaluation for testing our algorithm. For example, adapting our algorithm to DynamicCloudSim can more efficiently emulate the workflow execution and cloud failures.

References

[1] J. Montes, M. Zou, R. Singh, S. Tao, and M. Parashar, "Data-driven workflows in multi-cloud marketplaces," in *Cloud Computing (CLOUD), 2014 IEEE 7th International Conference on*, June 2014, pp. 168–175.

[2] A. Celesti, F. Tusa, M. Villari, and A. Puliafito, "How to enhance cloud architectures to enable cross-federation," in *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*, July 2010, pp. 337–345.

[3] 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. [Online]. Available: http://dx.doi.org/10.1186/2192-113X-1-15

[4] C. E. Shannon, "A mathematical theory of communication," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 5, no. 1, pp. 3–55, Jan. 2001. [Online]. Available: http://doi.acm.org/10.1145/584091.584093

[5] H. Hiden, S. Woodman, P. Watson, and J. Cała, "Developing cloud applications using the e-science central platform," *Royal Society of London. Philosophical Transactions A. Mathematical, Physical and Engineering Sciences*, vol. 371, p. 20120085, 2013.

[6] E. L. Rhys Lewis, *Introduction to Reliability Engineering*. Wiley, November 15, 1995.

[7] M. Hakem and F. Butelle, "Reliability and scheduling on systems subject to failures," in *Parallel Processing, 2007. ICPP 2007. International Conference on*, Sept 2007, pp. 38–38.

[8] A. Doğan and F. Özgüner, "Biobjective scheduling algorithms for execution time–reliability trade-off in heterogeneous computing systems*," *Comput. J.*, vol. 48, no. 3, pp. 300–314, May 2005. [Online]. Available: http://dx.doi.org/10.1093/comjnl/bxh086

[9] A. Dogan and F. Ozguner, "Matching and scheduling algorithms for minimizing execution time and failure probability of applications in heterogeneous computing," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 13, no. 3, pp. 308–323, Mar 2002.

[10] H. Kellerer, U. Pferschy, and D. Pisinger, *Knapsack Problems*. Berlin, Germany: Springer Verlag, 2004.

[11] Z. Wen, J. Cała, 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*, Dec 2014.

[12] D. Jang, J. U. Kang, A. Kruckman, J. Kudo, and S. J. Miller, "Chains of distributions, hierarchical bayesian models and benfords law," Tech. Rep., 2008.

[13] R. Prodan and M. Wieczorek, "Bi-criteria scheduling of scientific grid workflows," *Automation Science and Engineering, IEEE Transactions on*, vol. 7, no. 2, pp. 364–376, April 2010.

[14] J. Cała, Y. X. Xu, E. A. Wijaya, and P. Missier, "From scripted HPC-based NGS pipelines to workflows on the cloud," in *Procs. C4Bio workshop, co-located with the 2014 CCGrid conference*. Chicago, IL: IEEE, 2014.

[15] R. Van den Bossche, K. Vanmechelen, and J. Broeckhove, "Cost-optimal scheduling in hybrid iaas clouds for deadline constrained workloads," in *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*, July 2010, pp. 228–235.

[16] S. Song, K. Hwang, Y. kwong Kwok, and S. Member, "Risk-resilient heuristics and genetic algorithms for security-assured grid job scheduling," *IEEE Trans. Computers*, vol. 55, pp. 703–719, 2006.

[17] X. Qin and H. Jiang, "A novel fault-tolerant scheduling algorithm for precedence constrained tasks in real-no. 5, pp. 331–356, Jun. 2006. [Online]. Available: http://dx.doi.org/10.1016/j.parco.2006.06.006

[18] T. Xie and X. Qin, "Scheduling security-critical real-time applications on clusters," *Computers, IEEE Transactions on*, vol. 55, no. 7, pp. 864–879, July 2006.

[19] H. Fard, R. Prodan, and T. Fahringer, "A truthful dynamic workflow scheduling mechanism for commercial multicloud environments," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 24, no. 6, pp. 1203–1212, June 2013.

[20] H. Zo, D. L. Nazareth, and H. K. Jain, "Measuring reliability of applications composed of web services," *Proceedings of the Annual Hawaii International Conference on System Sciences*, pp. 1–10, 2007.

[21] S. Ran, "A model for web services discovery with qos," *SIGecom Exch.*, vol. 4, no. 1, pp. 1–10, Mar. 2003. [Online]. Available: http://doi.acm.org/10.1145/844357.844360