DOI 10.12694/scpe.v18i2.1289
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 18, Number 2, pp. 177–193. http://www.scpe.org

DR-SWDF: A DYNAMICALLY RECONFIGURABLE FRAMEWORK FOR SCIENTIFIC WORKFLOWS DEPLOYMENT IN THE CLOUD

KHADIJA BOUSSELMI*, ZAKI BRAHMI‡, AND MOHAMED MOHSEN GAMMOUDI§

Abstract. Workflows management systems (WfMS) are aimed for designing, scheduling, executing, reusing, and sharing workflows in distributed environments like the Cloud computing. With the emergence of e-science workflows, which are used in different domains like astronomy, life science, and physics, to model and execute vast series of dependents functionalities and a large amount of manipulated data, the workflow management systems are required to provide customizable programming environments to ease the programming effort required by scientists to orchestrate a computational science experiment. A key issue for e-science WfMS is how to deal with the change of the execution environment constraints and the variability and confliction of end users and cloud providers objectives for the execution of the same workflow or sub-workflow. They have to customize their management processes to insure the adaptability of the execution environment to the scientific workflows specificities, especially when dealing with large-scale (data, computing, I/O)-intensive workflows. In this paper, we propose a dynamically re-configurable framework for the deployment of scientific workflows in the Cloud (called DR-SWDF) that allows customizing the workflow deployment process according to a set of objectives and constraints of end users or cloud providers defined differently for the tasks or partitions of the same workflow. The DR-SWDF framework offers a K-means based algorithm that allows dynamically clustering the input workflows or sub-workflows in order to identify the most convenient techniques or algorithms to be applied for their scheduling and deployment in the cloud. The simulations results run on three examples large-scale scientific workflows show that our proposed framework can achieve better results than the use of a generic purpose approach.

Key words: workflow management system, scheduling, partitioning, provisioning, configuration, cloud computing, scientific workflows, K-means.

1. Introduction. The Cloud Computing environment offers multiple advantages for hosting and executing complex applications such as scientific workflows. The elasticity asset of the Cloud Computing resources helps such workflows to dynamically provision their compute and storage resources. Scientific workflows (called also e-Science workflows) are used in different domains such as astronomy, life science, physics, to model and execute vast series of dependents functionalities and a large amount of manipulated data [6]. Scientific workflow is a formal specification of a scientific process, which represents, streamlines, and automates the steps from dataset selection and integration, computation and analysis, to final data product presentation and visualization [16].

According to [17], the e-science workflow lifecycle, like traditional workflows, is defined by four phases, namely: the workflow composition and representation, the mapping of tasks to resources, the execution of the workflow and finally the recoding of metadata and provenance. These phases are usually included in a Scientific Workflow Management System (SWfMS) like Triana [24], Pegasus [25], and Taverna [23]. The goal of e-science workflow management systems is to provide a specialized programming environment to simplify the programming effort required by scientists to orchestrate a computational science experiment [17]. The SWfMS system supports the specification, modification, run, re-run, and monitoring of a scientific workflow using the workflow logic to control the order of executing workflow tasks [16].

A critical challenge of SWfMS for executing Scientific Workflows on the Cloud is how to efficiently allocate Cloud resources to the workflow tasks in order to optimize a predefined set of objectives. To do this, application schedulers employ different policies that vary according to the objective function: minimize the execution time, minimize cost, minimize the energy consumption, balance the load on resources used while meeting a fixed deadline or user budget constraints of the application, etc. In the Cloud Computing environment, the overall objective of task scheduling strategy is to guarantee the service-level agreements (SLAs) of allocated resources to make cost-efficient decisions for workload placement. Since task scheduling problem on Cloud Computing environments is NP-complete [19], several heuristics have been proposed to solve using Genetic Algorithms [38], Particle Swarm Optimization [39], Ant Colony Optimization [40], and Cat Swarm Optimization [5][6]. However, the problem is still challenging due to the dynamic nature of the Cloud Computing environment and to the variability and confliction of the defined scheduling objectives.

*Faculty of Sciences of Tunis, University of Tunis El Manar (khadija.bousselmi@gmail.com)
‡ISITCOM, University of Sousse (zakibrahmi@yahoo.fr)
§ISAMM, University of Mannouba, RIADI-GDL Laboratory, Tunisia (gammoudimomo@gmail.com)

The scheduling is particularly more challenging for large-scale scientific workflows as they may have thousands of tasks and data dependencies. Among the solutions, workflow partitioning is an approach to divide a workflow into several sub-workflows and then submit these sub-workflows to different execution sites (virtual clusters) [44]. Indeed, partitioning the original workflow into several fine-grained sub-workflows can ease the complexity of the workflow and increase its performances according to the partitioning objectives. The partitioning objectives can be varied and conflicting such as: optimizing the overall execution time of the workflow, minimizing the dependencies between the workflow partitions, minimizing the number of allocated resources, etc. For this reason, the problem of partitioning scientific workflows in the cloud computing environment is still an open research issue. Several approaches were proposed in the literature to partition scientific workflows according to one or multiple conflicting objectives [6][10][44].

Therefore, a major concern for a SWfMS is to be able to adapt his proposed features such as: the definition, scheduling, and resource provisioning to the scientific workflow's characteristics (computing, data or I/O intensiveness, number of tasks, input and output data size, etc.) and to the end users requirements and constraints (predefined deadline or budget, quality of service preferences, data placement constraints, etc.). The concept of adaptability has been introduced in the late 1990s and with the emergence of Adaptive Process Management (APM) systems. APM constitutes an evolutionary extension of the production workflow paradigm which intends to remedy the deficiencies related to dynamic process change [18]. According to [20], the adaptability of SWfMS can be defined at different layers, namely: the business context layer (defines the adaptability to a changing business context), the process layer (deals with changes to workflow models and their constituent workflow tasks), the resource layer (data model and organization changing) and the infrastructure layer (in case of system modifications or new technical settings).

In this work, we are interested in adaptable WfMS to the context modification as the scientific workflow can be considered as specific very complex workflows that are more difficult to be managed than traditional ones. Integrating a general-purpose WfMS in a specific business and organization context needs adaptation and reconciliation from both sides [20]. Dealing with context-different workflows with different requirements can impact the application performances of a workflow system. Thus, workflow systems should be prepared to adapt themselves to a range of different business and organization settings and also to a changing context [20]. Several SWfMS systems proposed in literature [23][24][25][26] offer adaptive management processes for the management and execution of scientific workflows in the Cloud by integrating the possibility of configuring these processes according to the user's requirements or the workflow's properties. However, local adjustments concerning a single task or a workflow partition are often neglected. Such local adjustments are very useful for a user to situate his/her work environment, including making decision in response to a special situation and on the basis of a variety of choices, reporting exceptional cases that are out of his/her responsibility, and so on [20]. More precisely, our research issue in this paper aims to dynamically configuring the workflow deployment process according to specific requirements of end users or cloud providers. We propose a dynamically configurable framework for the deployment of scientific workflows in the Cloud Computing environment inspired from the architecture of DR-FPGA (Dynamic Reconfigurable Field Programmable Gate Arrays) [8]. In our framework, we propose a K-means based algorithm that allows dynamically clustering the input scientific workflows in order to identify the most convenient techniques to apply for its deployment in the cloud. The proposed framework allows personalizing the deployment of the workflow at a fine-grained level (for specific tasks or partitions of the workflow) using different techniques at runtime and according to the input workflow's parameters, the user's requirements and the cloud provider's objectives. For instance, we consider only workflow partitioning, scheduling and resource provisioning as the main features of the SWfMS that can be dynamically configured; however, our framework is general purpose and can be extended to include more features.

This paper is an extension of the works originally reported in [5] and [6]. We use our proposed partitioning algorithm in [6] for the partitioning of scientific workflows and our proposed scheduling algorithm in [5] for their scheduling in the cloud.

The reminder of this paper is organized as follows. In Section 2, we present our mathematical model for scientific workflows deployment and, we introduce briefly the Dynamic Reconfigurable FPGA architecture. In Section 3, we present our proposed dynamic reconfigurable framework for scientific workflows deployment in the cloud. In Section 4, we exhibit our experimental setting and results. In Section 5, we present related works

and finally in section 6. we conclude.

2. Background. In this section, we will present the problem formulation for scientific workflows scheduling and deployment in the cloud. Then, we will present briefly the Dynamic Reconfigurable FPGAs as the architecture of our proposed framework in this paper is inspired from it.

2.1. Problem statement. In this section we will focus on modeling the scientific workflows in the Cloud computing environment guided. We will start by modeling the execution environment of the Cloud, modeling scientific workflows, characterizing QoS metrics and their evaluation techniques for workflows, defining and assessing the energy consumption of the Workflow. Finally, we will formulate our configurable optimization problem of workflows scheduling in the Cloud that we will use when scheduling the workflow in our proposed approach.

2.1.1. Modeling the execution environment. For the execution environment, we use the same modeling as proposed in our previous paper [6]. Indeed, the execution environment of scientific workflows can be organized as following: We dispose of a set of t data servers $S = \{S_i | i = 1, ..., t\}$ from the Cloud Computing environment. Each data server S_i is composed of m computing resources (Virtual machines or VM) such as $S_i = \{VM_i^k | k = 1, ..., m\}$. The Cloud providers offer different kinds of VM instances, adapted to different types of tasks (I/O, intensive computing, memory). We define a virtual machine VM_i by the quadruplet (R_i, C_i, Q_i, P_i) where:

- $R_i = \{R_i^k | k = 1, ..., d\}$ is the vector of types of resources of a virtual machine VM_i such as the processor, the internal memory SSD, the network device ... with d the maximum number of resource types of VM_i .
- $C_i = \{C_i^k | k = 1, ..., d\}$ is the vector of capacities of the resources of the virtual machine VM_i.
- $Q_i = (T_{exe}, C, A, R...)$ is the vector of the metrics of quality of service used to evaluate the performance of VM_i, such as $T_{exe} = \sum_{i=1}^{k} Dt_{in_i} * T_u$, where T_u is the time needed to process an input data of a unit task. $C = T_{exe} * C_u$ where C_u represents the cost per time unit incurred by using the virtual machine VM_i . A is the availability rate of VM_i and R its reliability value.
- $P_i = \sum_{k=1}^{d} P_i^k$ represents the energy consumption of the virtual machine (in Watts), defined according to its resource types where P_i^k defines the energy consumed by a resource type R_i^k of the VM for the execution of a unit task T per unit of time per unit data.

2.1.2. Modeling scientific workflows. As described in [6], we model a scientific workflow as a directed acyclic graph DAG which can be defined by the couple G = (V, E), where V=$\{T_1, ..., T_n\}$ is the vector representing the tasks of the workflow, with n the total number of tasks. E represents the functional links between tasks, and more specifically, the data dependencies between them, such as $(T_i, T_j) \in E$ if the output data of the task i are required for the execution of the task j. A workflow task is defined by the triplet T= (D, Dt_{in}, Dt_{out}) with:

- $Dt_{in} = \{Dt_{in} | i = 1, ..., k\}$ represents the amounts of data to be processed by each task of the workflow with k is the number of input datasets of the task, each dataset Dt_{in_i} need a minimum time T_i to be processed.
- $D = \{D_i | i = 1, ..., d\}$ is the vector of demands of the task in terms of resource capacities with D_i is the minimum capacity required of the resource type i to process the task and d the number of resource types required.
- $Dt_{out} = \{Dt_{out} | i = 1, ..., s\}$ represents the amounts of data generated after the completion of the workflow task.

2.1.3. Modeling the energy consumption of a workflow. Workflows execution in the Cloud involves allocating computing, network and storage resources used respectively for the execution of tasks, the transfer of data between dependent tasks and the storage of input or generated data [6]. The energy consumed by these allocated Cloud resources could then be divided into three components, namely: the processing energy, the storage energy and the data communication energy [6]. The processing energy is the energy consumed by the virtual machine to execute a specific task. This energy depends essentially on the virtual machine's configuration (processors number and capacities, memory, etc.), the time T_{exe} needed to execute a unit task

per unit data, the amount of data to be treated [6]. In this paper, we use the formula (2.1) proposed in [6] for the assessment of the processing energy of a virtual machine VM_i for the execution of a unit task T_j:

$$P_{exe_{ij}} = T_{exe_i} * P_i * Dt_{in_j} \tag{2.1}$$

with P_i is the VM's power consumed for the execution of a unit task per unit time per unit input data and Dt_{in} is the data amount to be treated by the VM for the execution of the task T_j. The storage energy represents the energy needed to store data on a permanent storage device (disk memory) [6]. This type of energy is usually negligible compared to the processor's energy [7].

The communication energy is the energy rate needed to transfer data from a virtual machine to another one using a network bandwidth [6]. The communication energy includes the data access energy (read/write) from DIMM memory (dual in-line memory modules) in source and destination VMs and the network power needed to transfer the data from one network to another E_{ij} [6]. Thus, we consider the formula (2.2) proposed in [6] to assess the energy consumed to transfer an amount of data Dt_{out_i} from a VM_i to a VM_j.

$$P_{transfer} = (Dt_{out_i} * E_{ij})/d_{ij} \tag{2.2}$$

where d_{ij} is the bandwidth rate between the source and destination machines.

In summary, the energy consumed for the execution of a workflow P_w is the sum of its total processing and data communication energy. This energy is described by the formula (2.3) used in our previous work [6]:

$$P_w = \sum_{(i=1)}^{n} \sum_{(j=1)}^{m} x_{ij} * (P_{ij} + \sum_{(k=i+1)}^{n} y_{ij} * Dt_{out_i} * E_{ik}/d_{ik}) \tag{2.3}$$

where: n is the total number of tasks of the workflow, m is the total number of VMs and x_{ij} is a boolean variable that is equal to 1 if the task i is assigned to the VM j and to 0 otherwise. y_{ij} is a boolean variable that is equal to 1 if there is a dependency between tasks i and j and to 0 if there is not.

2.1.4. Modeling and assessing the quality of service of a workflow. For the the modeling and the assessment of the quality of service of a workflow, we refer to our proposed formalism in our previous paper [5]. Indeed, we consider four quality metrics that can represent to our understanding the overall performances of the workflows, namely: the execution time, cost, resources availability and reliability. In the rest of this section we will detail the used equations for the assessment of the quality metrics for the workflow as well as its overall quality of service as described previously in [5]. As a workflow is composed of a set of tasks and data interdependencies between them, we consider that its execution time can be assessed by summing up the execution time of tasks and the data transfer time between them using the formula (2.4).

$$T_w = T_{execution} + T_{dataTransfer} \tag{2.4}$$

Likewise, as the Cloud providers (as Amazon or Google for example) fix a price for each of their basic services like executing a single operation on a specific computing resource or transferring a certain amount of data over network devices, we consider that the economic cost of executing a workflow can be assessed as the sum of the execution cost of the workflow tasks and the data transfer cost between dependent tasks using communication networks, as illustrated by the equation (2.5).

$$C_w = C_{execution} + C_{dataTransfer} \tag{2.5}$$

To assess the global values of the quality metrics of the workflow during its execution, we use aggregation functions for the different considered quality metrics. The formulas used for these aggregation functions are described in our previous work [5].

Once the quality metrics are assessed, we use the Simple Additive Weighting (SAW) normalization method, as in [5], to determine the value of the overall quality of service of the workflow. This latter Q_w can be evaluated via the equation (2.6).

$$Q_w = \sum_{i=1}^{K} w_i * q_i \tag{2.6}$$

where K is the number of considered quality metrics, q_i is one quality metric value assessed for the overall workflow (i.e. the execution time of the workflow) and w_i is the importance weight accorded to each quality metric with the sum of all importance weights is equal to one.

2.1.5. Configurable Scheduling model. Our ultimate objective in this work is to offer a personalized deployment process for each task or set of tasks of the workflow. This includes configuring the scheduling model of workflow to be adapted to the personalized scheduling objectives of its tasks, which implies defining multiple scheduling models for the same workflow. In this case we define a parameterized scheduling model that includes multiple scheduling objectives of both end users and cloud providers and to which we can add specific parameters to adapt it to a specific task or workflow.

The considered objectives in our configurable scheduling problem are: optimizing the quality of services described via the metrics of time, availability, and reliability, minimizing the cost (of execution and data transfer), and minimizing the energy consumption of the workflow, meeting a certain deadline, respecting a specific user budget. These metrics are conflicted for the following reasons:

- To execute quickly a workflow task, we should better use high performance Cloud resources (i.e. virtual machines) which are more expensive than others.
- High performance virtual machines use more resources (CPU, memory) which means that they consume more energy.
- Highly available and reliable resources are usually more expansive than others.

Thus, we modeled the workflows scheduling problem in the Cloud as a multi-objective optimization problem using the following linear representation:

$$\begin{cases} Energy : \text{minimize } w_1 P_w \\ Quality : \text{maximize } w_2 Q_w \\ Cost : \text{minimize } w_3 C_w \end{cases} \tag{2.7}$$

where: w_1 , w_2 and w_3 define respectively the importance weights of the energy consumption rate, the overall quality of service of the workflow and its overall cost to be defined when configuring the workflow scheduling. The quality of the workflow is, in turn defined to be configurable, by using a set of importance weights as illustrated by the equation (2.6). We can thus consider only the quality metrics that we want optimize.

2.2. Dynamic Reconfigurable FPGA (DR-FPGA). Field Programmable Gate Arrays (FPGAs) are one of the fastest growing parts of the digital integrated circuit market in recent times. They can be configured to implement complex hardware architectures. FPGA reconfiguration typically requires the whole chip to be reprogrammed even for the slightest circuit change [7]. The basic architecture of FPGA is based on an array of logic blocks connected through programmable interconnections as illustrated by the figure 2.1.

Dynamic reconfiguration means modifying the system when it is under operation. Reconfigurable computing utilizes hardware that can be adapted at run-time to facilitate greater flexibility without compromising performance. Reconfigurable architectures can exploit fine grain and coarse grain parallelism available in the application because of the adaptability [8].

Dynamically reconfigurable FPGA (DRFPGAs) systems can adapt to various computational tasks through hardware reuse [7]. Currently there are very large capacity DRFPGAs which contain many highly parallel fine grain parallel processing power, and the ability to define high bandwidth custom memory hierarchies offers a compelling combination of flexibility and performance. In addition FPGAs are able to adapt to a real-time processing. We can integrate functional improvements without wasting time receiving hardware or modifying the layout of the circuit. The exploitation of parallelism provides performance advantage over conventional microprocessors [8]. As illustrated by the figure 2.1, the red and the green solid lines represent an example of two kinds of connection ways respectively that allow processing differently the same input data to generate different outputs. The definition of the connection path to be performed can be done in real time according to a certain number of parameters (input data, output data, performance preferences, etc).

The granularity of the reconfigurable logic of FPGA is defined as the size of the smallest functional unit (configurable logic block, CLB) that is addressed by the mapping tools [8]. High granularity, which can also be known as fine-grained, often implies a greater flexibility when implementing algorithms into the hardware.

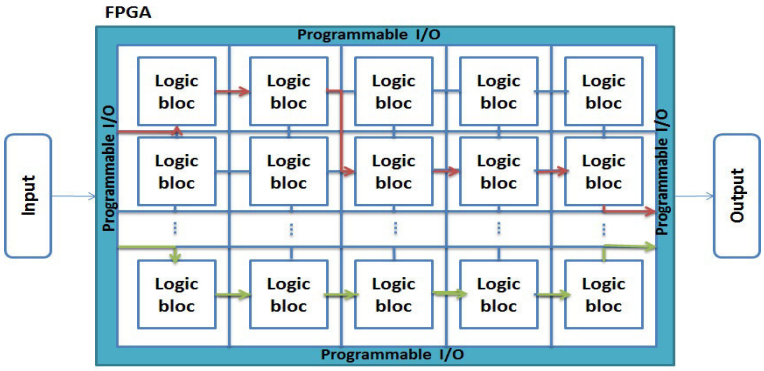

FIG. 2.1. *Architecture of DR-FPGA*

However, there is a penalty associated with this in terms of increased power, area and delay due to greater quantity of routing required per computation [8]. In the contrary, small granularity on coarse-grained architectures are intended for the implementation of algorithms needing word-width data paths. As their functional blocks are optimized for large computations and typically comprise worldwide arithmetic logic units, they will perform these computations more quickly and with more power efficiency than a set of interconnected smaller functional units.

3. DR-SWDF: Dynamic Reconfigurable Scientific Workflows Deployment Framework. In this section, we describe our proposed dynamic reconfigurable framework for scientific workflows deployment that allows dynamically configuring and executing these workflows in the Cloud environment. The main advantage of our proposal is that, for the same input workflow, we can apply different scheduling, partitioning or resource provisioning techniques to each sub-group of interdependent tasks of the workflow at design time and at runtime. We consider that each portion of tasks the workflow can have different characteristics (such as the tasks types, input and output datasets, etc), different scheduling objectives (prefixed budget, deadline, performance constraints, energy consumption rate, etc) and other constraints (such as the placement of input/output data). These characteristics can be defined during the configuration of the workflow before its deployment. Our framework allows also dynamically clustering the input workflow or sub-workflow according to its description by using a K-means clustering algorithm in order to identify the most convenient techniques to be applied for its deployment in the cloud.

The main components of our workflows deployment framework are, namely: the configuration, partitioning, scheduling, provisioning, and the deployment component. Each of the offered components can implement different algorithms and/or techniques. Each algorithm is better suited to the requirements of a specific category of scientific workflows. The figure 3.1 illustrates the main functional components of our proposed DR-SWDF. We inspired the architecture of our framework from that of the DR-FPGA presented in the previous section by replacing each logic bloc by a software component (algorithm, technique, tool, etc). Like DR-FPGAs, the execution process of a scientific workflow using our proposal follows a directed path relying different components starting from the configuration step to the deployment of the workflow. More specifically, the DR-SWDF framework allow to redefine the processing path for a specific partition of the workflow at design time or at runtime in order to use more suitable algorithms to the sub-workflow's characteristics (tasks type, scheduling constraints, input/output data size, etc) for its partitioning, scheduling or provisioning. As an example, the paths illustrated by red and green solid lines in the figure 3.2 represent two different processes to be applied when executing a scientific workflow for two of its partitions. The dynamic reconfiguration step can be performed dynamically before the execution of each partition of the workflow by examining its characteristics and considering personalized techniques for its processing which allow improving the performances of the overall workflow. We consider that the reconfiguration step can be performed without affecting the performances of the overall workflow execution as it can be planned well in advance.

As illustrated by the figure 3.2, the functional process of our proposed framework follows a directed path. For

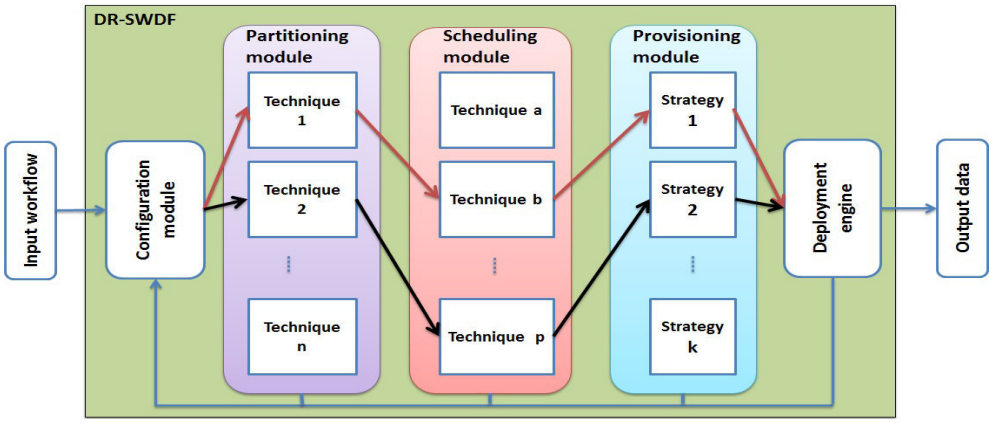

FIG. 3.1. *Overview of our proposed DR-SWDF framework*

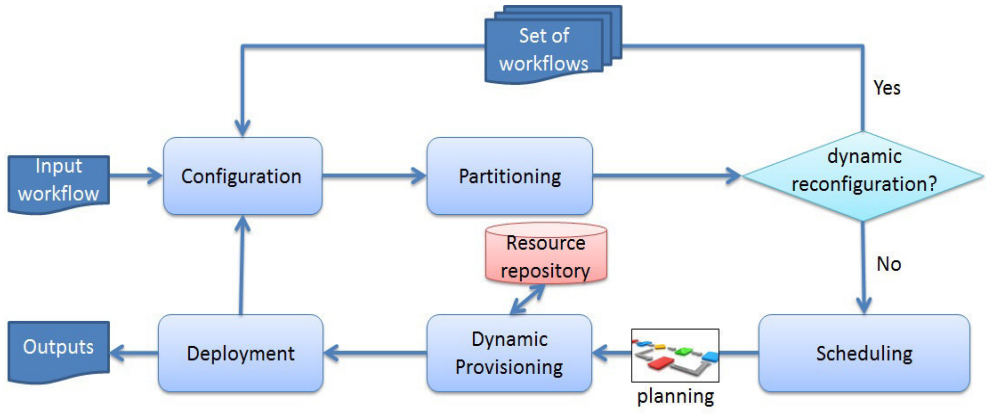

FIG. 3.2. *Functional overview of DR-SWDF framework*

a specific input workflow, the DR-SWDF precedes to the configuration of the workflow deployment by defining, firstly, the workflow type (as computing, I/O or memory intensive) using a K-means clustering algorithm. Based on the workflow type, number of tasks and the number of available resources, the configuration algorithm determines the most convenient partitioning, scheduling and provisioning techniques or algorithms to apply if it's possible. After the partitioning step, if the dynamic reconfiguration option is active (which is active by default), the set of generated workflow partitions are returned to the configuration step to be dynamically reconfigured as distinct workflows with their own characteristics and input/output data. If the dynamic reconfiguration option is not active (which means that the workflow is sufficiently fine grained to be processed at once), the deployment process moves to the scheduling and provisioning of the workflow using the selected techniques during the configuration step.

The framework design is generic and can be extended by adding others components and/or algorithms for the deployment of workflows in the Cloud environment like data reuse, data cleanup and fault tolerance optimization. For instance, we consider only partitioning, scheduling and provisioning of scientific workflows with using two different algorithms for each function to illustrate the advantages of our approach. In the following, we will detail the role of each component and the used algorithms or techniques for each one.

3.1. Workflow Configuration and Dynamic Reconfiguration. The configuration is a key feature of our approach. It allows defining the techniques or strategies to be used for the deployment of the input workflows. As in some cases sub-parts of the workflow might present different characteristics from the overall workflow [41], it is critical then reconsider these parts as separate workflows that requisite different techniques

to be deployed in the cloud, and here where the dynamic reconfiguration is needed.

The dynamic reconfiguration can be performed for the overall scientific workflow at once if all its parts have the same characteristics or for a specific partition of the workflow. Its main objective is to reconsider the characteristics of the input sub-workflow for the choice of the most suitable algorithms or strategies to be used for its deployment in the cloud. These characteristics include the following metrics: the workflow type (computing intensive, memory intensive, I/O intensive or a combination of these types), the number of tasks, the estimated makespan and energy consumption of each one, the quality constraints of end users (budget, makespan, etc), the cloud provider's constraints (limited number of VMs, storage memory, etc), and the input and the output data sizes.

Based on the workflow size, model and structure, we defined a set of metrics to initially classify the input workflow or sub-workflow of our framework as computing intensive or data intensive or I/O intensive, etc. Then, we used a clustering algorithm to be able to automatically classify the input workflows, especially those which include different types of sub-workflows.

For instance, we consider the equation (3.1) to identify whether the input workflow is data intensive or not.

$$\sum_{i=1}^{n} \frac{(D_{in}^{i} + D_{out}^{i})}{n} \geq Limit_{dataSize} \tag{3.1}$$

where: n is the overall number of tasks, D_{in}^{i} is the input data size of the task i and D_{out}^{i} is its output data size which is not an input data for another task. $Limit_{dataSize}$ is the threshold of data size beyond it we can consider the workflow as data intensive.

We can identify also a computing intensive workflow by the fact that its tasks need a long time to be processed. Hence, we defined the equation (3.2) to automatically define a computing intensive workflow.

$$\max_{1 \leq i \leq j} \frac{t_{exe_i}}{k} \geq T_{limit} \tag{3.2}$$

where: k is the number of the workflow tasks that should be performed sequentially, j is the number of tasks that can be executed in parallel, t_{exe_i} is the estimated execution time of the task i and T_{limit} is the execution time threshold beyond it a workflow can be considered as computing intensive. As it is one of the most popular clustering algorithms, we opted for the K-means clustering algorithm [42] to classify the input workflows or sub-workflows and map them to the most relevant partitioning, scheduling and provisioning techniques that fit their characteristics. K-means aims to partition an input set of N points into K clusters by finding a partition such that the squared error between the empirical mean of a cluster and the points in the cluster is minimized. The squared error metric and more details about the K-means algorithm could be found in [42].

The main steps of K-means algorithm are as follows [42]:

1. Select an initial partition with K clusters; repeat steps 2 and 3 until cluster membership stabilizes.
2. Generate a new partition by assigning each pattern to its closest cluster center.
3. Compute new cluster centers.

The K-means algorithm requires three user-specified parameters: number of clusters K, cluster initialization, and distance metric. The number of clusters in this work is limited to three defining the type of the workflow intensity (I/O, memory, computing). The definition of initial clusters is accomplished using the equations (3.1) and (3.2) determining a first classification of the workflows. Then, we use the graph distance measure defined in [55] to evaluate the similarity degree between the DAG graphs of the input workflows. In fact, the distance of two non-empty graphs G_1 and G_2 can be defined by the equation (3.3) [55]. Intuitively, the larger distance of two graphs is, the more similar the two graphs are. The use of this similarity metric for the clustering of scientific workflows is more convenient as it is able to capture the (structural) difference between the workflows more than other vector space based distances such as the Euclidian distance.

$$d(G_1, G_2) = 1 - \frac{|mcs(G_1, G_2)|}{max(|G_1|, |G_2|)} \tag{3.3}$$

where $mcs(G_1, G_2)$ denotes the maximal common subgraph of two graphs G_1 and G_2 and $|G_i|$ is the number of arcs in the graph G_i. The algorithm 1 describes the steps of the dynamic re-configuration algorithm proposed for our DR-SWDF framework.

Algorithm 1: Workflow Dynamic Re-configuration Algorithm

```
input  : Var w;                                                  ▷ the input workflow
         Var budget;                                             ▷ The end user's budget constraint
         Var makespan;                                           ▷ The workflow completion time constraint
         Var vm − number;                                        ▷ number of available VMs
         Var memory − limit;                                     ▷ The cloud provider's constraint for maximum storage memory
         Var cpu − limit;                                        ▷ The cloud provider's constraint for cpu capacity limit
         Var dynamic − reconfiguration − option;                 ▷ (0: not active, 1: active)
output: Var P;                                                   ▷ The index of the partitioning algorithm 'P'
         Var S;                                                  ▷ The index of the scheduling algorithm 'S'
         Var Pr;                                                 ▷ The index of the provisioning algorithm 'Pr'
begin
    dynamic − reconfiguration − option = 1;
                                          ▷ set the dynamic-reconfiguration-option to active by default
    intworkflow − type = K − means − cluster(W);
                                          ▷ define the type of the workflow by applying the K-means algorithm
    intp − tasks = count − parallel − tasks(W);
                                          ▷ determine the number of tasks that may run in parallel
    if p − tasks ≤ vm − number then
        dynamic − reconfiguration − option = 0;
                                          ▷ disable the dynamic reconfiguration option
    else
        P = partitioning − technique − choice(workflow − type);
                                          ▷ set the partitioning technique corresponding to the workflow type
    if dynamic − reconfiguration − option == 1 then
        partitions[] = workflow − partition(P, W);
                                          ▷ apply the partitioning algorithm and get partitions
        forall the workflow − partition ∈ partitions[] do
            Workflow − Dynamic − Re − configuration − Algorithm(workflow − partition);
                                          ▷ for each workflow partition, re-apply the algorithm from the beginning
    else
        S = scheduling − technique − choice(workflow − type);
                                          ▷ set the convenant scheduling technique
        set − objective − function(S, String[]parameters);
                                          ▷ Set the objective function parameters such as the Budget and Makespan
        set − provisioning − constaints(vm − number, memory − limit, cpu − limit);
                                          ▷ set the provisioning constraints
        Pr = scheduling − technique − choice(workflow − type);
                                          ▷ set the convenant provisioning technique
```

The configuration component should, also, insure the planning of the deployment order of the workflow partitions and their input/output data sets, and handle errors or execution problems. It should be able to decide if the execution process should be reprogrammed or canceled in case of error.

3.2. Workflow Partitioning. The partitioning function aims to split the workflow composed of a large number of tasks into several sub-workflows in order to achieve a high level of parallelism for the execution of the workflow. Many algorithms were proposed in literature for workflows partitioning such as [6][9][10]. Each algorithm can have a different partitioning objective such as minimizing the overall cost of the workflow (by minimizing the data transfer rate between distant resources) or minimizing its makespan (by maximizing the number of tasks that can be performed in parallel). These two last objectives are conflicting because minimizing the data transfer rate implies fostering the use of the same cloud resources or those situated in the same server, while optimizing the makespan implies using powerful and highly available cloud resources regardless of their location. To determine which algorithm is more suitable to a specific workflow or sub-workflow, we consider the following characteristics: the workflow type, its tasks number and their estimated execution time, with the input and output data sizes. Our DR-SWDF framework analyses these data and chooses the most convenient technique to be considered at the configuration step.

For example, for the Montage workflow [2], which is a computing and data intensive workflow designed

to manage science-grade mosaics of the sky, we can foster a partitioning technique that allow to minimize the data movement between partitions for the overall workflow at the configuration step such as in [6]. But at the dynamic reconfiguration of the workflow partitions we can opt for the optimization of its processing time by using a partitioning algorithm that insures a high level of parallelism such as in [9]. Hence, we can optimize the makespan of the workflow and minimize its cost (through minimizing the data transfer rate over communication networks) thanks to the use of two different partitioning techniques for the same input workflow.

In our previous work [6], we proposed a Workflow Partitioning for Energy minimization (WPEM) algorithm that allows reducing the network energy consumption of the workflow and the total amount of data communication while achieving a high degree of parallelism. The main idea of the WPEM algorithm is to apply the technique of max flow/min-cut to recursively partition the workflow into two sub-workflows such as the data communication amount between them is minimal. The experimental results show that WPEM allows reducing remarkably the network energy consumption of both the memory and data intensive workflows, however it doesn't perform better than the [9] for computing intensive workflows [6]. In [9], the authors propose a new algorithm WPRC (Workflow Partition Resource Clusters) for scheduling Scientific Workflows in the Cloud environment. The WPRC algorithm partitions the workflow in order to achieve the highest level of parallelism and thus, optimizing the cost and time of execution of the workflow and the tasks of each partition are, then, assigned to the cluster selected according to their execution priority.

Hence to cover computing, data and memory intensive workflows, we will consider only the partitioning algorithms in [6] and [9] as they offer promoting results, each for a specific kind of workflows. However, our framework is flexible and can integrate other partitioning algorithms.

3.3. Workflow Scheduling. The third function of our proposed framework is the scheduling of the input workflow or sub-workflow according to predefined objectives (such as optimizing the cost and/or the makespan of the workflow, optimizing the QoS of the involved Cloud resources or minimizing their energy consumption rate, etc) as illustrated in the section 2.5. The scheduling of a workflow tasks consists to affect every generated sub-workflow (or workflow partition) from the partitioning step to a Cloud data server and schedule the partition tasks using its virtual machines. The workflow scheduling problem is to carry out a mapping of these tasks to VMs that optimizes its scheduling objectives and constraints. These objectives can be defined at the configuration step for the overall workflow or personalized at the dynamic reconfiguration of the workflow partitions. The utility of defining different scheduling objectives for some partitions of the workflow can be explained by the fact that the tasks of these partitions may be more important than others or manipulate critical data that should be placed within certain constraints.

Being an NP-complete problem, several optimization algorithms were proposed in literature to deal with the issue of multi-objective scheduling of workflows like [5][11][12]. In this context, we implement among others our proposed optimization algorithm (EPCSO) in a previous work [5] which is an enhancement of the Parallel Cat Swarm Optimization algorithm [21]. This algorithm allows resolving the multi-objective optimization problem corresponding to the scheduling objectives of the workflow's user and offers betters results compared to others optimization heuristics such as [11][12]. EPCSO allows optimizing the overall QoS of the workflow by considering one or more metrics like the execution time, the data transmission time, and the cost of the workflow, the cloud resources availability, reliability and the data placement constraints. A second optimization algorithm considered by our framework is that proposed in [12], the authors propose a new PSO-based algorithm to solve the scheduling problem of workflows in Cloud Computing environments. The objective of the scheduling problem is to optimize the makespan and the cost of the workflow. The results of the proposed algorithm show its ability to reduce the overall cost of the workflow and maintain an even distribution of work packages among the allocated resources. We consider for this moment only these two optimization algorithms as they cover the most conflicting scheduling objectives that a user of a scientific workflow can express but it remains possible to add other heuristics in the future.

3.4. Resources Provisioning. Resource provisioning is a key step of the workflow deployment process. It aims to insure efficient use of the Cloud resources which is a common goal for both workflow users and cloud providers. The efficiency in virtual resource provisioning has direct influence on the Quality of Service (QoS) of IaaS clouds [13]. For end users, using the right resources for the right tasks will allow saving their time and money by applying the 'pay-what-you-go' model of the Cloud Computing. For Cloud providers, resource

provisioning is finality in itself and an important way to save the energy consumption of their cloud resources. The strategy to be used for resources provisioning depends on several factors such as the type of the workflow (computing or data or I/O intensive, etc), its tasks number and their characteristics (the required software configuration, the estimated execution time, the input/output data, etc), and the needed or generated data size and placement constraints. These factors can be different for some tasks of same workflow, thus, our proposed DR-SWDF offers the possibility to consider a different provisioning strategies for each workflow partition at the dynamic reconfiguration step.

In this scope, we implemented two different provisioning techniques from literature: the first one is designed to the provisioning of VMs [14] and the second one is focalized on the provisioning of storage resources [15]. We have chosen these two techniques as they insure the provisioning of both types of involved cloud resources in the process of scientific workflow execution in the cloud. The goal of VM provisioning is to provide sufficient resources to meet the level of QoS expected by end-users. For this purpose, most cloud provides deliver a set of general-purpose VM instances with different resource configurations. For example, Amazon EC2 provides a variety of VM instance types with different amounts of resources. In [14], Jing et al. modified the genetic algorithm with a fuzzy multi-objective evaluation in order to search for a large solution space by considering the conflicting objectives such as power consumption minimization, total resource wastage, and thermal dissipation costs. The authors designed, also, a genetic algorithm for VM placement. The simulation results show that the algorithm outperforms existing traditional greedy approaches in terms of the number of utilized hosts and performance degradations.

Otherwise, as scientific workflows can be very complex, one task might require many datasets for execution; furthermore, one dataset might also be required by many tasks. If some datasets are always used together by many tasks, we say that these datasets are dependents on each other [15]. The goal of storage resources provisioning is to insure an efficient use of the available storage resources in terms of cost and energy consumption by avoiding excessive data access. To do this, we used our proposed algorithm in [5], in which, we modeled the problem of storage resources selection in the cloud as a multi-objective optimization problem based on the criteria of: availability, cost and their approximation to the virtual machines on which the tasks will be executed.

3.5. Workflow deployment. The workflow deployment is not the final step of the workflow management process according to our proposal. It becomes a central component to which the workflow partitions or tasks ready to be executed are submitted. The execution process is started once the process of dynamic reconfiguration is achieved. To do this, after the partitioning of each workflow or sub-workflow that should respect the execution order of the workflow tasks, every partition follows its own process including task scheduling, resources provisioning, before its submission to the workflow deployment engine as an autonomous workflow with its own input datasets, allocated resources, start time and deadline if defined. If the deployment of a specific partition fails due to a specific reason such as unavailable resource or input dataset access failure, a notification should be sent to the configuration component. In this case, the configuration component should decide according to the failure reason whether it will reprogram the execution of the workflow partition or cancel the overall workflow execution. The monitoring of the whole execution process of the workflow is insured by the configuration component which centralizes all information about the workflow (partitions, input/output data, execution result, resources available, etc).

4. Simulations results. We use the WorkflowSim [35] simulation environment, on which we implemented our proposed framework. WorkflowSim extends the CloudSim [36] simulation toolkit by introducing the support of workflow preparation and execution with an implementation of a stack of workflow parser, workflow engine and job scheduler. It supports a multi-layered model of failures and delays occurring in the various levels of the workflow management systems.

We consider three workflows that we configured to be computing and data intensive workflows, notably the Montage [2], Cybershake [3] and Epigenomics [4] workflows. These workflows were chosen because they represent a wide range of application domains and a variety of resource requirements [37]. The Montage project is an astronomy application that delivers science-grade mosaics of the sky [2]. The Montage workflow is I/O intensive and the figure 4.1 illustrates a simplified structure of it. We used the Montage Workflow with 4006 tasks as in [6]. The CyberShake workflow is used to calculate Probabilistic Seismic Hazard curves for several

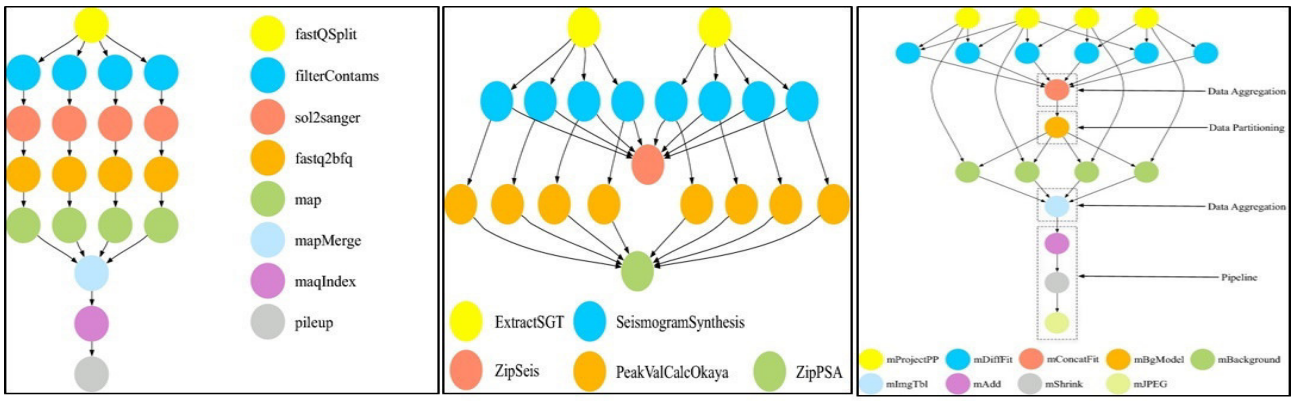

FIG. 4.1. *A simplified representation of the Epigenomics, Cybershake and Montage Wokflows respectively*

geographic sites in the Southern California area [3]. The CyberShake workflow is memory intensive. A portion of this workflow is illustrated by the figure 4.1 containing 2252 tasks. Finally, the Epigenomics project maps short DNA segments collected with high-throughput gene sequencing machines to a previously structured reference genome [4]. The Epigenomics workflow is CPU intensive. A simplified structure of this workflow is described by the figure 4.1. The used Epigenomics Workflow contains 4000 tasks. We used the execution traces of the considered workflows from [1] to generate the values of unitary task's execution time and data communication amounts between dependents tasks of these workflows.

As we don't have enough informations about the different techniques offered by the studied SWfMS like the partitioning algorithm type and objective, we compared our proposed approach to a "generic" one. The "generic" approach corresponds to the use of one technique of partitioning, scheduling and provisioning for the overall workflow among those implemented in our framework and that for all the considered workflows. We use the same simulation environment for the "generic" approach as in our proposed framework. We run four scenarios with different conflicting scheduling objectives:

1. Scenario 1: optimizing both the makespan and the energy consumption
2. Scenario 2: optimizing both the energy consumption and the cost,
3. Scenario 3: optimizing both the cost and the makespan,
4. Scenario 4: optimizing the energy consumption and the overall quality of service of the workflow.

In each scenario, we compare the use of our approach to the generic approach for which we use the most appropriate techniques among those implemented in our framework to apply for the overall workflow. The results of our simulations are detailed below.

For the scenario 1, the overall objective is optimizing the makespan and the energy consumption of the workflow. To satisfy this objective using the generic approach, we used our proposed partitioning algorithm in [6], the scheduling algorithm in [5] and the provisioning strategy of [14]. For our proposed approach, we used for each partition different techniques among those implemented in our framework that corresponds more to the partition type.

The simulation results in figure 4.2 show that our proposed approach allows reducing both the makespan and the energy consumption of all the tested workflows compared to the generic approach. This reduction is due to the use of techniques that are more suited to the workflow tasks/partition rather than the overall workflow.

In the same way, our proposed approach offers better results than the generic one for the second scenario by allowing reducing both the cost and the energy consumption of the tested workflows as illustrated by the figure 4.3.

For the scenario 3, our objective is to minimize both the cost and the makespan of the workflow which depicts the most important objectives of end users. By using our proposed approach, we can achieve better values than using the generic approach designed for all workflow without considering their type and tasks particularities (see figure 4.4).

In the final scenario, our objective is to optimize the overall quality of service of the workflow and its energy

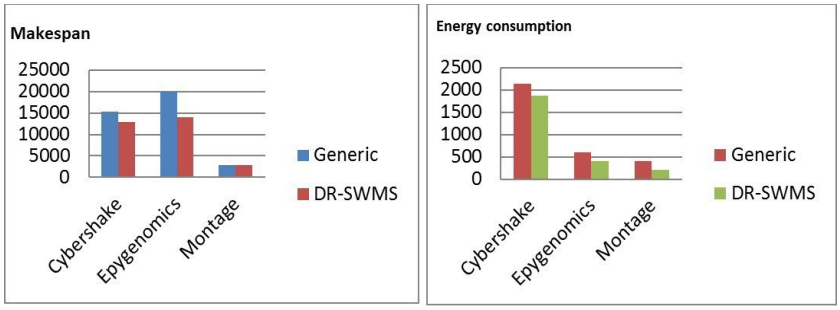

FIG. 4.2. *Simulation results for the scenario 1*

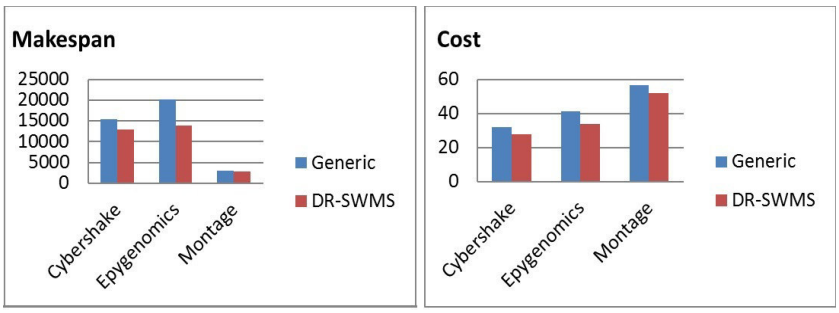

FIG. 4.3. *Simulation results for the scenario 2*

consumption. We assessed the overall quality of service of the workflow by using the metrics of time, availability, and reliability of all the resources involved in the workflow execution. Thus, the considered objectives are conflicting as a high highly available resource is a resource that consumes more energy. The figure 4.5 illustrates the simulation results of our proposed approach compared to the generic approach and confirms that our approach offers better quality of service rate while using less energy than the generic one for all the tested workflows and thus due to the use of more adaptable scheduling objectives for each partition of the workflow, in addition to the use of multiple provisioning and partitioning algorithms for the same workflow. In summary, our proposed framework offers to the end users and cloud providers the occasion to personalize the process of workflow deployment in the cloud by dynamically reconfiguring the partitions or tasks of the workflow as independent workflow and using different techniques for their deployment and conserving the data dependency between them. Our simulation results show that this approach can achieve better results than the use of a generic one within the considered scheduling as in the tested scenarios.

5. Related works. Many SWfMS were proposed in literature, such as Kepler [22], Taverna [23], Triana [24], Pegasus [25], ASKALON [26], SWIFT [27] have demonstrated their ability to help domain scientists on scientific computing problems by synthesizing data, application and computing resources [28]. Other research works that dealt with the scientific workflows deployment in the Cloud proposed their systems as "Software as a service" such as in [31][30][29]. The challenges dealt with in the e-science workflows systems/works can be categorized into three aspects: composition, execution and the workflow composition and learn. The workflow composition aspect involves the features of workflow definition, medialization, usability, etc. The execution aspect includes workflow scheduling, partitioning, resource provisioning, optimization, and workflow adaptability. Finally, the learn aspect includes analyzing the data provenance and results of the workflow execution in order to enhance knowledge learn and the workflow performances.

In this paper, we focus on the execution aspect of the SWfMS, and more precisely on the problems related to adaptability of the execution process to the change of the business context of the scientific workflows. The execution process of such workflows includes the following aspects: the partitioning, scheduling, deployment, resource provisioning, and data storage management especially for data intensive workflows.

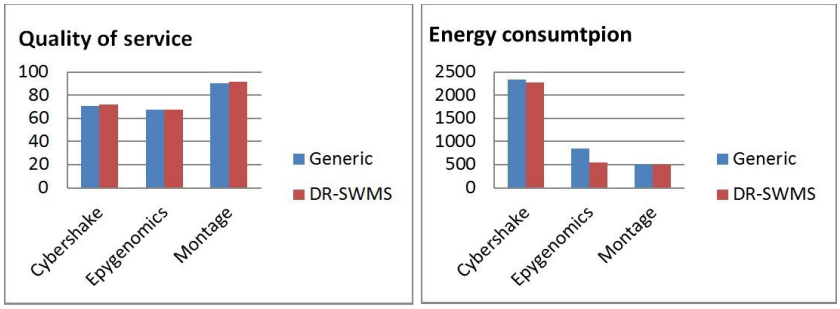

FIG. 4.4. *Simulation results for the scenario 3*

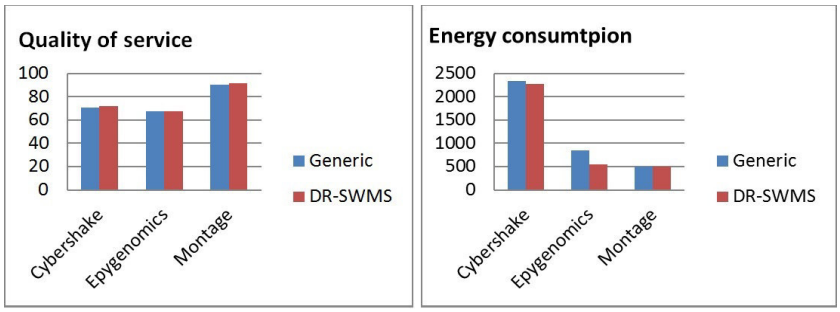

FIG. 4.5. *Simulation results for the scenario 4*

Most of the proposed SWfMS include the data driven adaptability (they deal with the workflow's data changing during the execution process). For example, Triana uses Aspect Oriented Programming (AOP) techniques to perform the workflow rewriting, effectively creating sub-workflows that execute and feed back into the main workflow in order to allow dealing with newly integrated or modified datasets. Pegasus, also, allows, redefining the workflow during the execution by changing the input data or the resources to be used. Likewise, Kepler workflows can modify themselves during execution. Another aspect of adaptability of workflow systems is computation unfolding, one task or sub-workflow could actually need to be iterated or parallelized during the execution based on different conditions. To enable this adaptation aspect, some workflow systems like Pegasus and Swift support abstract workflows [17].

For data intensive workflows, the overall objective of SWfMS is to optimize the data manipulation, transfer and storage within the minimum cost and time. For example, Pegasus technique consists to move data to local or remote locations where scientific applications employed in the workflows will be deployed. If one workflow contains applications deployed on different sites, there have to be multiple data movements during the workflow execution [28]. This technique allows optimizing the overall makespan of the workflow but when the data sizes are very large, times for data movements will be significant and result in very inefficient workflow execution time and cost since data transfer in the cloud is paying. In Triana, in the case of task-based workflow, the user can designate portions of the workflow as compute intensive and Triana will send the tasks to the available distributed resources for execution.

The capability of a SWfMS to allow defining different scheduling objectives and applying multiple techniques for the tasks of the same workflow according to the importance of these tasks or to other parameters (like task type, input/output data size, etc.) is another important feature that SWfMS should consider especially in the era of Big Data emergence. In this context, Pegasus performs a mapping of the entire workflow, portions of the workflow, or individual tasks onto the available resources executed and provides an interface to a user defined scheduler limited to some feature such as the makespan and the execution cost and includes four basic scheduling algorithms, namely: HEFT [33], min-min, round-robin, and random. The Askalon system, designed to support task-level workflows, has a rich environment for mapping workflows onto resources. However, its scheduler

makes full-graph scheduling of scientific workflows, using one of the implemented scheduling algorithms such as HEFT.

Resource provisioning adaption is another crucial requirement for SWfMS. Provisioning is slightly more complex than queuing in that it requires users to make more sophisticated resource allocation decisions [34]. Existing provisioning strategies could be classified into two kinds: static and dynamic and are mainly cost or deadline-constrained. In static provisioning the application allocates all resources required for the computation before any jobs are submitted, and releases the resources after all the jobs have finished [34]. In dynamic provisioning resources are allocated by the system at runtime. This allows the pool of available resources to grow and shrink according to the changing needs of the application [34]. Pegasus implements both static and dynamic provisioning to allow cost and deadline-constrained scientific workflows deployment in the Cloud [32]. Other SWfMS like Taverna and Triana include dynamic provisioning algorithms that are applicable for the overall resources implied for the workflow execution. None of the existing SWfMS does offer customizable resource provisioning techniques that correspond to different types of workflows.

6. Conclusions. In this paper, our research objective is to dynamically cluster scientific workflows and ensure their scheduling and deployment according to the specific requirements of end users or cloud providers. To do this, we consider each single partition or task of the workflow as a separate workflow and define a specific execution process for it. We propose DR-SWDF, a dynamically configurable framework for the deployment of scientific workflows in the Cloud that allows using different techniques at runtime according to the input workflow's parameters, the user's requirements and the cloud provider's objectives. The simulations results run on three examples of data and computing intensive workflows show that our proposed framework can achieve better results than the use of a generic one within different conflicting scheduling objectives such as optimizing the energy consumption, the cost or the makespan of the workflow.

In the future, we expect to extend our framework in order to implement more techniques for the considered features and include new features like the data provenance management, data reuse, data cleanup and fault tolerance optimization.

REFERENCES

[1] https://pegasus.isi.edu/workflow_gallery/

[2] http://montage.ipac.caltech.edu/

[3] https://scec.usc.edu/scecpedia/CyberShake

[4] http://epigenome.usc.edu.

[5] Bousselmi, K., Brahmi, Z., & Gammoudi, M. M., *QoS-Aware Scheduling of Workflows in Cloud Computing Environments*, In 2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA) (pp. 737–745). IEEE.

[6] Bousselmi, K., Brahmi, Z., & Gammoudi, M. M., *Energy efficient partitioning and scheduling approach for Scientific Workflows in the Cloud*, In Services Computing (SCC), 2016 IEEE International Conference on (pp. 146–154). IEEE.

[7] Kwiat, K. A., *Dynamically reconfigurable fpga apparatus and method for multiprocessing and fault tolerance*, U.S. Patent No 5, 931–959, 3 Aug 1999.

[8] Bondalapati, K. and Prasanna, V., *Reconfigurable computing systems*, Proceedings of the IEEE, 2002, vol. 90, no 7, p. 1201–1217.

[9] Bagheri, R., & Jahanshahi, M., *Scheduling Workflow Applications on the Heterogeneous Cloud Resources*, Indian Journal of Science and Technology, 8(12).

[10] Tanaka, M., and Osamu T., *Workflow scheduling to minimize data movement using multi-constraint graph partitioning*, Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012). IEEE Computer Society.

[11] Pandey, S., Wu, L., Guru, S. M., & Buyya, R., *A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments*, In 2010 24th IEEE international conference on advanced information networking and applications (pp. 400–407). IEEE.

[12] Bilgaiyan, S., Sagnika, S., & Das, M., *A multi-objective cat swarm optimization algorithm for workflow scheduling in cloud computing environment*, Intelligent Computing, Communication and Devices, Advances in Intelligent Systems and Computing, 308, Springer India 2015.

[13] Lu X, Wang H, Wang J, Xu J, Li D., *Internet-based virtual computing environment: beyond the data center as a computer*, Future Generation Computer Systems, 2013, 29(1): 309–322

[14] Xu, J., & Fortes, J. A., *Multi-objective virtual machine placement in virtualized data center environments*, In Green Computing and Communications (GreenCom), 2010 IEEE/ACM Int'l Conference on & Int'l Conference on Cyber, Physical and Social Computing (CPSCom) (pp. 179–188). IEEE.

[15] Yuan, D., Yang, Y., Liu, X., & Chen, J., *A data placement strategy in scientific cloud workflows*, Future Generation Computer Systems, 26(8), 1200–1214.
[16] Lin, C., Lu, S., Lai, Z., Chebotko, A., Fei, X., Hua, J., & Fotouhi, F., *Service-oriented architecture for VIEW: a visual scientific workflow management system*, In Services Computing, 2008. SCC'08. IEEE International Conference on (Vol. 1, pp. 335–342). IEEE.
[17] Deelman, E., Gannon, D., Shields, M., & Taylor, I., *Workflows and e-Science: An overview of workflow system features and capabilities*, Future Generation Computer Systems, 25(5), 528–540.
[18] Guenther, C. W., Reichert, M., & van der Aalst, W. M., *Supporting flexible processes with adaptive workflow and case handling*, In Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2008. WETICE'08. IEEE 17th (pp. 229–234). IEEE.
[19] L. Guo, S. Zhao, S. Shen, and C. Jiang., *Task scheduling optimization in cloud computing based on heuristic algorithm*, Journal of Networks, 7(3):547–553, 2012.
[20] Han, Y., Sheth, A., & Bussler, C., *A taxonomy of adaptive workflow management*, In Workshop of the 1998 ACM Conference on Computer Supported Cooperative Work.
[21] Tsai, P. W., Pan, J. S., Chen, S. M., Liao, B. Y., & Hao, S. P., *Parallel cat swarm optimization*, In Proceedings of the seventh international conference on machine learning and cybernetics, Kunming, China, 1:3328–3333,2008.
[22] Ludscher B., Altintas I., Berkley C., Higgins D., Jaeger E., Jones M., Lee E., Tao J., & Zhao Y., *Scientific workflow management and the Kepler system*, Concurr Comput Pract Exp 18 (10): 1039–1065.
[23] Oinn T., Addis M., Ferris J., Marvin D., Senger M., Greenwood M., Carver T., Glover K., Pocock M.R., Wipat A., & Li P., *Taverna: a tool for the composition and enactment of bioinformatics workflows*, Bioinformatics 20 (17): 3045–3054, Oxford University Press, London.
[24] Taylor I., Shields M., Wang I., & Harrison A., *The Triana workflow environment: architecture and applications*, In: Taylor I, Deelman E, Gannon D, ShieldsM(eds)Workflows for e-Science. Springer, New York, pp 320–339.
[25] Deelman E, Mehta G, Singh G, Su M, Vahi K, *Pegasus: mapping large-scale workflows to distributed resources*, In: Taylor I, Deelman E, Gannon D, Shields M (eds)Workflows for e-Science. Springer, New York, pp 376–394.
[26] Fahringer T, Jugravu A, Pllana S, Prodan R, Seragiotto Jr, C, Truong H, *ASKALON: a tool set for cluster and Grid computing*, Concurr Comput Pract Exp 17 (2–4): 143–169, Wiley InterScience.
[27] Zhao Y., Hategan M.,Clifford B., Foster I., von Laszewski G., NefedovaV., Raicu I., Stef-Praun T.,Wilde M., *Swift: fast, reliable, loosely coupled parallel computation*, Proceedings of 2007 IEEE congress on services (Services 2007), pp 199–206.
[28] Yang, X., Wallom, D., Waddington, S., Wang, J., Shaon, A., Matthews, B., ... & Vasilakos, A. V., *Cloud computing in e-Science: research challenges and opportunities*, The Journal of Supercomputing 70 (1), 408–464.
[29] Hosni, E., & Brahmi, Z., *OaaS Based on Temporal Partitioning with Minimum Energy Consumption*, Procedia Computer Science 96, 540–549.
[30] Esteves, S., & Veiga, L., *WaaS: Workflow-as-a-Service for the Cloud with Scheduling of Continuous and Data-intensive Workflows*, The Computer Journal 59 (3), 371–383.
[31] Wei, Y., & Blake, M. B., *Adaptive service workflow configuration and agent-based virtual resource management in the cloud*, In Cloud Engineering (IC2E), 2013 IEEE International Conference on (pp. 279–284). IEEE.
[32] Malawski, M., Juve, G., Deelman, E., & Nabrzyski, J., *Cost-and deadline-constrained provisioning for scientific workflow ensembles in IaaS clouds*, In Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (p. 22). IEEE Computer Society Press.
[33] Topcuoglu, H., Hariri, S., & Wu, M. Y., *Performance-effective and low-complexity task scheduling for heterogeneous computing*, IEEE transactions on parallel and distributed systems 13 (3), 260–274.
[34] Juve, G., & Deelman, E., *Resource provisioning options for large-scale scientific workflows*, In eScience, 2008. eScience'08. IEEE Fourth International Conference on (pp. 608–613). IEEE.
[35] Chen, W., & Deelman, E., *Workflowsim: A toolkit for simulating scientific workflows in distributed environments*, In E-Science (e-Science), 2012 IEEE 8th International Conference on (pp. 1–8). IEEE.
[36] Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A., & Buyya, R., *CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms*, Software: Practice and Experience 41 (1), 23–50.
[37] Chen, Weiwei, and Ewa Deelman., *Partitioning and scheduling workflows across multiple sites with storage constraints*, Parallel Processing and Applied Mathematics. Springer Berlin Heidelberg, 2011, 11–20.
[38] M. Rahman, X. Li, and H. N. Palit., *Hybrid heuristic for scheduling data analytics workflow applications in hybrid cloud environment*, In Proceedings of the 25th IEEE International Symposium on Parallel and Distributed, ser. IPDPS Workshops. Anchorage (Alaska) USA, Anchorage (Alaska) USA, pp. 966–974, May 2011.
[39] Rodriguez, M. A., & Buyya, R., *Deadline based resource provisioning and scheduling algorithm for scientific workflows on clouds*, IEEE transactions on Cloud Computing, 2014.
[40] R. Achary, V. Vityanathan, P. Raj, and S. Nagarajan, *Dynamic job scheduling using ant colony optimization for mobile cloud computing*, Advances in Intelligent Systems and Computing, 321, Springer International Publishing Switzerland 2015.
[41] Ramakrishnan, L., & Plale, B., *A multi-dimensional classification model for scientific workflow characteristics*, In Proceedings of the 1st International Workshop on Workflow Approaches to New Data-centric Science (p. 4). ACM.
[42] Jain, A. K., *Data clustering: 50 years beyond K-means*, Pattern recognition letters 31 (8), 651–666.
[43] Bunke, H., & Shearer, K., *A graph distance metric based on the maximal common subgraph*, Pattern Recognition Letters, 19 (3-4): 255–259, 1998.

[44] Chen, W., & Deelman, E., *Integration of workflow partitioning and resource provisioning*, In Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012) (pp. 764–768). IEEE Computer Society.

Edited by: Dana Petcu
Received: Jan 13, 2016
Accepted: May 31, 2017