CloudSim4DWf: A CloudSim-Extension for Simulating Dynamic Workflows in a Cloud environment

Fairouz Fakhfakh
ReDCAD Laboratory
FSEGS, University of Sfax
B.P. 1088, 3018 Sfax, Tunisia
fairouz.fakhfakh@redcad.org

Hatem Hadj Kacem
ReDCAD Laboratory
FSEGS, University of Sfax
B.P. 1088, 3018 Sfax, Tunisia
hatem.hadjkacem@redcad.org

Ahmed Hadj Kacem
ReDCAD Laboratory
FSEGS, University of Sfax
B.P. 1088, 3018 Sfax, Tunisia
ahmed.hadjkacem@fsegs.rnu.tn

***Abstract*—The resources provisioning for workflow applications has become one of the most difficult challenges in the cloud. It consists in taking an appropriate decision when mapping tasks to resources while meeting QoS requirements. The need to modify workflows while they are being executed has become a major requirement to deal with unusual situations and evolution. Therefore, it is necessary to implement a provisioning policy which takes into account dynamic changes of workflow, while satisfying some performance criteria defined by the user. Simulation tools are an efficient solution to evaluate the performance of cloud applications. They offer a free environment that can mimic the behavior of a real cloud environment. Nevertheless, existing simulators are based on static application models. In this paper, we introduce CloudSim4DWf, which extends the existing CloudSim simulator with a new resource provisioning policy for dynamic workflows. The different experiments that we present show the efficiency of our tool. .**

***Keywords*—Cloud computing, resources provisioning, workflow, simulation, dynamic changes**

I. Introduction

Workflows technology has gained great attention in recent years for the development of applications in different scientific fields such as astronomy, bioinformatics and physics [6]. Their interest comes mainly from the need to build upon legacy codes that would be too costly to rewrite. It is commonly modeled by a Directed Acyclic Graph (DAG) in which graph nodes represent tasks and graph edges represent data dependencies among the tasks.

Recently, the ability to modify workflows while they are being executed has become a major requirement especially for a long-running applications. This functionality is essential to deal with exceptional situations and failure that may occur during the execution of a workflow instance. We cite, for example, the case when the execution time of a task exceeds a certain value, vehicle accident, technical problems, etc. In addition, dynamic adaptation can handle new situations to deal with the rapid changes demanded by the dynamic nature of the marketplace. We cite for example the strategy shifts, changes in the customer behavior and the market evolution [19] [11]. In fact, the economic success of enterprises depends increasingly on their ability to respond to changes in their environment in a quick and flexible way. The changes of a workflow may be at the functional level (adding or deleting tasks, etc.) or at the non-functional level (such as changing time constraints).
Cloud computing infrastructure has gained an increasing popularity to run workflow applications. In fact, it offers flexible capacity on computation and storage and less installation expense [18]. The elasticity is the main characteristic of cloud which enables such dynamic workflow to be enacted more efficiently. In fact, it facilitates the changing of resource quantities at runtime by scaling up or down to meet demand [12].

In this context, the resource provisioning for dynamic workflows still one of the real problems that needs further investigations. Evaluating the performance of these provisioning strategies in a real infrastructure under different conditions is a hard problem. In fact, cloud environments are subject to variations that make it difficult to reproduce the environment and conditions of an experiment [9]. Also, the financial cost and time required by these experiments are often very high. Then, using simulation frameworks is a more efficient solution that enables controlled experimentation, reproducible results and comparison of many solutions in similar environments.

Currently, the available simulation solutions which are specific to the cloud (such as CloudSim [3] and GreenCloud [13]) make it possible to model a cloud environments and to simulate different workloads running on them. However, they are based on static application models and use provisioning algorithms that cannot handle dynamic changes caused by some events that can be triggered at runtime.
This paper introduces a new simulator called *CloudSim4DWf* (CloudSim for Dynamic Workflow) which aims to evaluate resources provisioning policies for dynamic workflow applications in the cloud. CloudSim4DWf extends CloudSim [3] with some modules to ensure the resources management when an event that needs structural adaptation actions is triggered at runtime.

The rest of this paper is organized as follows: In section II, we present the latest research works in the field of cloud computing simulators. Section III gives a brief overview of CloudSim tool. In section IV, we highlight the key components of our simulation tool CloudSim4DWf. Section V introduces a detailed description of its design and implementation. Simulation results and evaluation are discussed in section VI. Finally, the last section concludes and outlines areas for our future research.

II. Related work

Recently, the use of simulation tools in the field of cloud computing has become very widespread. A significant number of simulators has been developed. A detailed state of the art that presents the existing simulaors in cloud has been presented in our previous work [8]. In the following, we review some related works.

MDCSim [15] is a discrete event simulator. It helps to model unique hardware characteristics of different components of a datacenter such as servers, communication links and switches which are collected from different dealers. *MDCSim* tool is a commercial product. It is not available for public download. However, the others are under open source.

GreenCloud [13] uses an accurate and sophisticated network model. It is used for the simulation of packet-level and energy consumption, which are not the focus of this work.

iCanCloud [16] is a new simulator for modeling cloud infrastructures. It offers a graphical interface to interact with the simulator. The major drawback of this simulator is that it supports modeling and simulation of only EC2 (Elastic Compute Cloud 2) instances.

The simulation tools that we have already presented are not able to isolate the multi-layer service abstractions (IaaS, PaaS and SaaS) required by the cloud environments. However, CloudSim is a generalized simulation tool which allows modeling, simulation, and experimentation of cloud infrastructures and application services. It provides a controlled environment which is easy to set-up to test the performance of applications. In addition, it can represent many types of clouds (public, private, hybrid and multi-cloud environments). Users define workloads by creating cloudlets instances, which are submitted and processed by VMs (virtual machines) deployed in the cloud [4].
One of the key aspects of CloudSim is that it is easily extensible. It has originated several extensions in the literature [5][9][20][1][14]. In fact, many attempts have been made to add new functionalities to it based on different requirements. We cite afterwards some of these extensions.

WorkflowSim is an enhanced version of CloudSim that provides a layer of workflow management in cloud [5]. This layer introduces support for workflows simulation and scheduling algorithms. However, this tool is based on a rigid workflow structure and it is tightly bound to the scientific workflow management systems (SWfMS) Pegasus [7].

Garg et al. [9] present *NetworkCloudSim* which incorporates resources for modeling applications and datacenter network behaviors. This simulator adds to *CloudSim* the ability to represent and manage network equipment depending on the changes in servers state.

DynamicCloudSim [1] extends *CloudSim* to simulate instability and dynamic performance changes in VMs during runtime. It can determine whether a task succeeds or fails.

FTCloudSim is another extension to the *CloudSim* framework proposed by Zhou et al. [20][21]. It supports simulation of fault tolerance mechanisms. An extensible interface is offered to help researchers in implementing new mechanisms easily. Moreover, it can trigger failure events to test the performance of these mechanisms. After execution, it generates information about some metrics to demonstrate the advantages and shortcomings of the mechanism.

FederatedCloudSim introduces new enhancements on top of CloudSim basic platform to simulate many federated cloud scenarios. Moreover, it adds packages for testing SLA-aware scheduling algorithms.

The choice of *CloudSim* over other simulators in this article has been motivated by its more mature code base, the important number of available extensions and its generic cloud model.
To the best of our knowledge, none of the available simulation platforms have attempted to simulate events which need dynamic adaptation actions at workflow instance level and its impact on the resources provisioning.

III. CloudSim Overview

CloudSim [4] is a well-known cloud simulator that can represent different types of clouds. It was developed by a leading research group in grid and cloud computing called CLOUDS Laboratory at the university of Melbourne in Australia. It is based on GridSim [2].
CloudSim is an event-driven simulation tool, that is to say, all components of the simulation maintain a message queue and generate messages, which they pass along to other entities. It can instantiate multiple datacenters. Each one consists of storage servers and physical host machines, which host several VMs running several tasks (named cloudlets in CloudSim). It can simulate the assignment and execution of a given workload on a cloud infrastructure [3]. The flow of communication among core CloudSim entities is as follows:
- At the beginning of a simulation, each datacenter entity registers its information to the Cloud Information Service (CIS) registry.
- The datacenter broker (DCB) acting on behalf of users queries the CIS Registry for the information of datacenters.
- The CIS registry responds by sending a list of the available datacenters.
- The DCB requests the characteristics of the available datacenters (DC).
- The DCB asks the concerned DC to create the required VMs.
- Once the VMs are created, the DCB sends cloudlets to DC for execution. If cloudlets finish their execution, a message is sent to the DCB.
- At the end of the simulation, the VMs will be destroyed.

IV. Proposed extension: CloudSim4DWf

As already shown, CloudSim implements the user's application model as simple objects describing computational requirements for the application. It does not simulate dependencies between cloudlets. Then, it is not dedicated to a workflow application. Also, it does not handle how to react to changes at application level during simulation. We mention, for example, the need to add or to substitute cloudlets at runtime. In order to ensure an efficient resources provisioning for a dynamic workflow, we extend CloudSim by three modules. The first one offers a graphical user interface (GUI) which enables users to

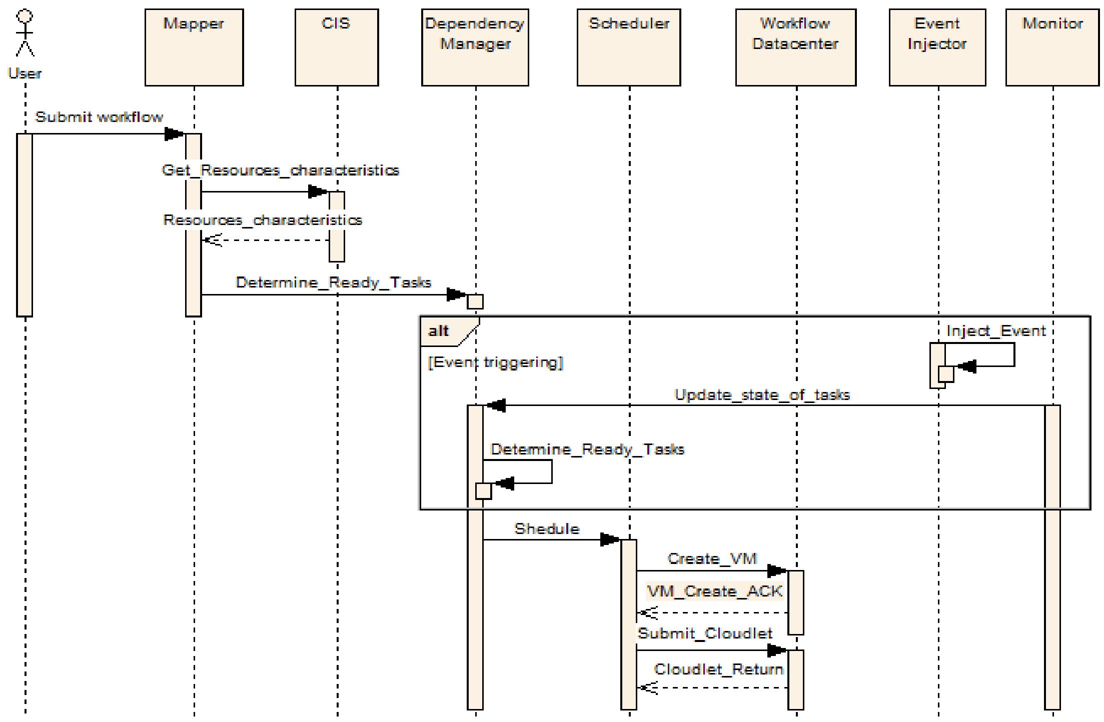


Fig. 1. Sequence Diagram: Communication between CloudSim4DWf Entities

manage the different types of VMs and provide the inputs needed for simulation. The second module aims to trigger some events in a random way at runtime. These events need a dynamic adaptation of the workflow application. The third module aims to ensure an efficient resources provisioning for a dynamic workflow.
In order to support workflow applications, we suppose that a workflow can be represented according BPMN [1](Business Process Modeling Notation) semantics. In this work, we consider three basic types of change operations: ADD, REPLACE and DELETE as illustrated in Figure 2. ADD inserts a new

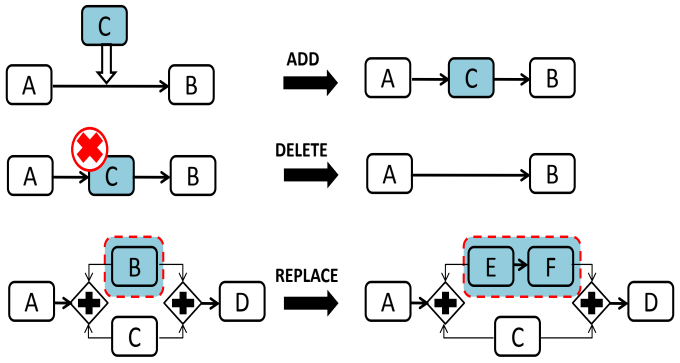
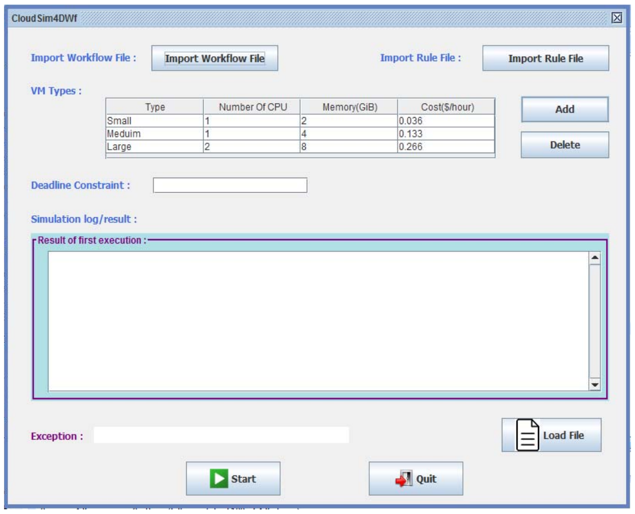


Fig. 2. Change operations

fragment to the workflow model between a predecessor task ‘A’ and a successor task ‘B’. REPLACE replaces an existing fragment with a new one, whereas DELETE allows removing an existing fragment.
Figure 1 depicts the communication between the different entities of the three modules of CloudSim4DWf. The bloc ”Alt” illustrates the case of triggering an exceptional event. In the following, we detail the role of each entity of the three modules.

A. Graphical user interface (GUI)


CloudSim4DWf

Import Workflow File : Import Workflow File

Import Rule File : Import Rule File

VM Types :

Type	Number Of CPU	Memory(GiB)	Cost($/hour)
Small	1	2	0.036
Meduim	1	4	0.133
Large	2	8	0.266

Add
Delete

Deadline Constraint :

Simulation log/result :

Result of first execution :

Exception :

Load File
Start
Quit

Fig. 3. GUI of our tool

Our GUI enables users to import the workflow description file and the adaptation rules defined for the same workflow (see Figure 3). In Listing 1, we illustrate an example of an adaptation rule which consists in adding two tasks when an event is raised at task T_2. Each added task has these characteristics (length, file size, etc.). A user can also manage

[1]http://www.bpmn.org/

the different types of VMs (adding or deleting VM types). After specifying the deadline constraint value, he can start the simulation. Thus, a log file containing the simulation traces is generated.

Listing 1. Adaptation rule

```
<Rule id ="R1" name="Rule_1">
        <Event name="Raised_on_T2" task_exception="2"/>
        <Condition name=""/>
        <Action value= "Add_Task" nb_tasks="2">
            <Before id= "5" />
            <After  id= "2"/>
                <Characteristic_task>
                  <Task id ="8" lenght="315895" fileSize=
                  "3012" outputSize="3166657"/>
                  <Task id ="9" lenght="337509" fileSize=
                  "4095" outputSize="3366547"/>
                </Characteristic_task>
        </Action>
</Rule>
 ..
```

B. Events injection module

We have added two components which are responsible for triggering and monitoring the simulation events that can be raised at runtime:

- **Event injector**: At random moments of time, it generates some events which need adaptation actions such as adding tasks, deleting tasks or substituting tasks at runtime.
- **Monitor**: It collects the triggered events and returns them to the "Dependency manager" in order to update the list of the ready tasks. So, the scheduler have to take into account the adaptation actions. In fact, it is necessary to reassign the unexecuted tasks to resources in order to meet the deadline constraint. Let's take the example of the workflow presented in Figure 4. We suppose that an event is raised just after the execution of task T_3. This event needs to add two tasks T_8 and T_9. In order to meet the deadline constraint of the workflow, the scheduler recomputes the mapping solutions of unexecuted tasks (for example: T_5, T_6 and T_7) and those added (T_8 and T_9) and reassign them to the appropriate types of VMs. In this example, we assume that T_5 and T_6 have not yet started their execution. But, we can have other cases.

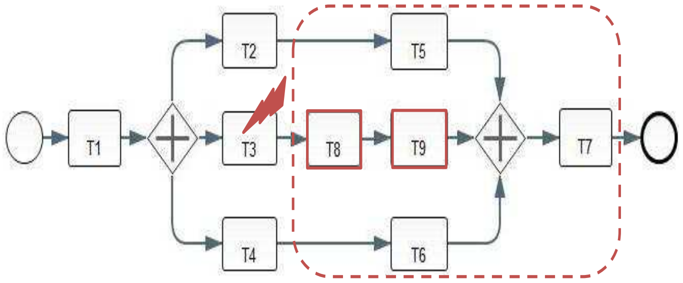


Fig. 4. Workflow example after adaptation

C. Resources provisioning module

Our provisioning algorithm (see Algorithm 1) is dynamic as it reschedules tasks at runtime. It proceeds by marking all tasks of workflow as unexecuted (line 2). After that, it computes the mapping solutions for all workflow tasks (line 3). Then, the set of ready tasks are submitted to its assigned resources for execution (line 4) and the tasks statuses are updated (line 5). If an event which needs an adaptation action is raised at runtime, it is necessary to update the mapping solutions for tasks which have not begun their execution yet (line 8). These steps are repeated until all the workflow tasks complete their execution.

Algorithm 1 Dynamic provisioning algorithm

```
Require: $W$: Workflow application, $D$: Deadline constraint of W,$A$: mapping algorithm
Ensure: Resources provisioning for the workflow taking into account events triggered at runtime and QoS requirements.
 1: procedure DYNAMIC_PROVISIONING($W$, $D$, $A$)
 2:     Mark all tasks of workflow as unexecuted ($Unexecuted$[]:=all workflow tasks)
 3:     Compute the initial mapping solutions using the algorithm "A"
 4:     Dispatch ready tasks to its assigned resources
 5:     Update the status of tasks
 6:     while $Unexecuted$[] is not empty do
 7:         if an event $E$ is triggered then
 8:             Compute the new mapping for unexecuted tasks using the algorithm "$A$"
 9:         end if
10:         Submit ready tasks for execution
11:         Update the task statuses
12:     end while
13: end procedure
```

In the following, we describe the main components of this module.

- **Mapper**: This component aims to map each task to the appropriate VM type for execution. Each workflow has a deadline constraint (D) which defines the time limit for its execution. In order to determine the mapping result of tasks to resources, we have developed an algorithm based on the meta-heuristic particle swarm optimization (PSO) [17].
- **Dependency manager**: It handles the data dependencies between tasks. It ensures that a task may only be released to the scheduler when all of its parent tasks have completed execution successfully.
- **Scheduler**: It asks the workflow datacenter to create the required VMs.

V. Implementation of CloudSim4DWf

CloudSim4DWf extends CloudSim's functionality with the introduction of some features that can handle the resources provisioning when an event is triggered at runtime. The events that we have considered need structural adaptation actions. The main parts of this extension are depicted in the class diagram of Figure 5. Classes presented in white color are part of CloudSim, however, those shown in grey color are the added classes that enable the addition of the new features.
In the remainder of this section, we provide some details related to the basic classes of CloudSim, which constitute the

building blocks of the simulator. Then, we describe the new classes of CloudSim4DWf.

A. CloudSim classes

- **SimEntity**: This is an abstract class which represents a simulation entity that can send messages to other entities. Also, it processes the received messages as well as triggers and handles events.
- **SimEvent**: This class presents a simulation event which is passed between two or more entities. It stores some information about an event such as type, init time, finish time, IDs of the source and destination entities, tag of the event and data that have to be passed to the destination entity.
- **Datacenter**: This class models the services, at infrastructure level, offered by cloud providers. The arrow from Datacenter to SimEntity indicates that the former is a subclass of the latter.
- **DatacenterCharacteristics**: This class contains configuration information of datacenter resources.
- **DatacenterBroker**: This class models a broker which acts as a mediator between users and service providers according to users' QoS requirements. Also, it deploys service tasks across clouds.
- **CloudInformationService**: The CIS class provides resource registration, indexing, and discovering capabilities.

B. Extension classes

- **EventInjector**: This added class extends the SimEntity class. It is responsible for inserting some events at random moments of time during simulation.
- **MyEvent**: This class extends the SimEvent class. The simulated events in our work are created by EventInjector entity. They need some adaptation actions of the workflow application at runtime. Their tag type is called "INJECT_EVENT". In CloudSim, the tags are used to identify the kind of action to be taken.
- **EventMonitor**: It is an extended class of SimEntity. It receives notifications of the different events triggered at runtime as well as the adaptation actions that must be performed.
- **Mapper**: The Mapper has a global view of the whole workflow (all the tasks and their dependencies). The concrete implementation of this class incorporates our provisioning algorithm which determines the resource appropriate to each task of the workflow.
- **Dependency manager**: It is an extended class of SimEntity. It determines the list of the ready cloudlets. Then, it sends them to the Scheduler with the VMs that must be created.
- **Scheduler**: The Scheduler class extends CloudSim's DatacenterBroker class. A broker in CloudSim terminology acts on behalf of the user for the creation and destruction of VMs and submission of cloudlets to the VMs. The extended Scheduler entity can handle the submission of job which are appropriate for workflow model (instead of cloudlet).
- **WorkflowDatacenter**: The WorkflowDatacenter class extends the CloudSim's Datacenter class. The difference between them is that the former is appropriate for workflow model.

In our extension, we have considered that each cloudlet is characterized by its status which can be:

- *Unmapped*: The cloudlet has not been mapped to a resource yet.
- *Mapped*: The cloudlet is assigned to a resource but has not been submitted for execution.
- *Submitted*: The cloudlet has been submitted to its adequate resource for execution.
- *Running*: The cloudlet is running, knowing that a cloudlet can begin its running only if it is ready. That means that all cloudlet predecessors have finished their execution.
- *Finished*: The cloudlet has finished execution and the result is ready for use or transfer.

VI. Performance evaluation

In this section, we present the experiments conducted in order to evaluate the efficiency of CloudSim4DWf simulator in provisioning resources for a dynamic workflow application. In fact, our goal is to notice its behavior at the provisioning level when an event which requires an adaptation action is raised during execution. We compare the performance of our simulator to WorkflowSim as it is the only one which supports workflow applications.

A. Experimental setup

In our experimental setup, we defined an IaaS provider offering a single datacenter and five different types of VMs. The VM configurations are based on current Amazon EC2[2] offerings (see Table I). We used a VM billing interval of one hour and a boot time of 60 seconds.
Experiments are conducted with a library of realistic workflows which are used in scientific domains [10]. We have selected workflows representing different types of applications: Montage, LIGO, SIPHT and CyberShake. For each type of these workflows, four kinds of DAG with different number of tasks are defined.

TABLE I. Configurations and prices of VMs

VM type	Number of CPUs	Memory (GiB)	Cost ($ per hour)
micro	1	1	0.018
small	1	2	0.036
meduim	2	4	0.072
large	2	8	0.134
xlarge	4	16	0.491
4xlarge	16	64	1.966
10xlarge	40	160	4.914

[2]http://aws.amazon.com/fr/ec2/

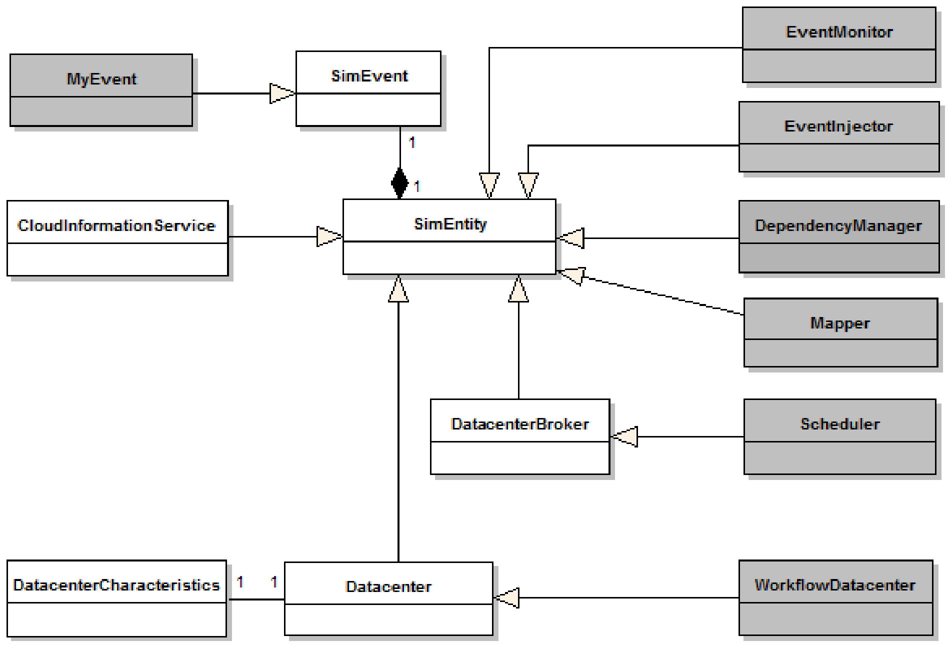


Fig. 5. Class diagram of CloudSim4DWf

B. *Experimental result*

In the following, we evaluate the performance of our simulation tool by studying its overhead and financial cost.

1) Overhead evaluation: Our simulator uses dynamic provisioning algorithm which consists in remapping some tasks to resources at runtime. In fact, it begins by affecting all workflow tasks to resources for execution. Whenever an event is raised at runtime, it becomes necessary to remap unexecuted tasks to the adequate resources in order to satisfy the deadline constraint. Then, an overhead is induced while rearranging the resources. In this experiment, we compute the average time required for the resources selection in the case of two scenarios: an optimal execution and an adaptive execution.

In the first scenario, we assume that no event which requires an adaptation action is raised at runtime. To do so, we use CloudSim4DWf simulator while disabling the events injection module. So, the selection of resources appropriate to each task is performed before the workflow execution.
In the second and third scenario, some events which need dynamic adaptation actions are triggered at runtime. Then, our simulator needs to recompute the mapping solutions for tasks which have not begun execution yet. We suppose that the raised events are generated with a uniform distribution. The maximum number of the events does not exceed one fourth the number of workflow tasks. In the second scenario, we use our algorithm based on the meta-heuristic optimization technique PSO. However, in the third scenario, we use an exact solution which determines the most optimal solution using the CPLEX framework [3].

We report the measured computing time for the three scenarios of execution by varying the number of workflow tasks (see Figure 6: Montage (a), LIGO (b), SIPHT (c) and CyberShake (d)). The overhead represents the percentage of additional computing time of an adaptive execution compared to the optimal execution. As illustrated in the figure, the overhead of the second scenario does not exceed 15% for all cases. Nevertheless, using an exact solution requires a higher computing time especially when the number of tasks increases.

2) Financial cost evaluation: To evaluate the efficiency of our simulator in term of financial cost, we present two scenarios:
In the first one, we have used our CloudSim4DWf simulator and we have varied the rate of the events triggered by the event injector entity at runtime {15%, 30%, 45%, 60%, 75%, 90%}. We suppose that each event needs to add, delete or substitute some tasks at runtime. The triggered events and the necessary adaptation actions are generated randomly. We have taken into account the average of the remapping time caused by the triggered events.
In the second scenario, we have used WorkflowSim simulator while using the same provisioning algorithm (based on PSO). Since this simulator can not handle the events raised at runtime, the assignment of tasks to resources is performed before the workflow execution. In this case, we need to take into account all events which can be triggered during execution. To do so, we consider the maximum of total execution time which can be taken by workflow tasks to ensure that resources are sufficient for execution. The simulation results displayed in Figure 7 are the averages of 300 experiments (Montage (a), LIGO (b), SIPHT (c) and CyberShake (d)). These results show a considerable gain in financial cost when using our CloudSim4DWf simulator.

VII. Conclusion and Future Work

In this paper, we have introduced CloudSim4DWf, a new simulation tool for dynamic workflow applications in the cloud environment. CloudSim4DWf injects some events that

[3]http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/

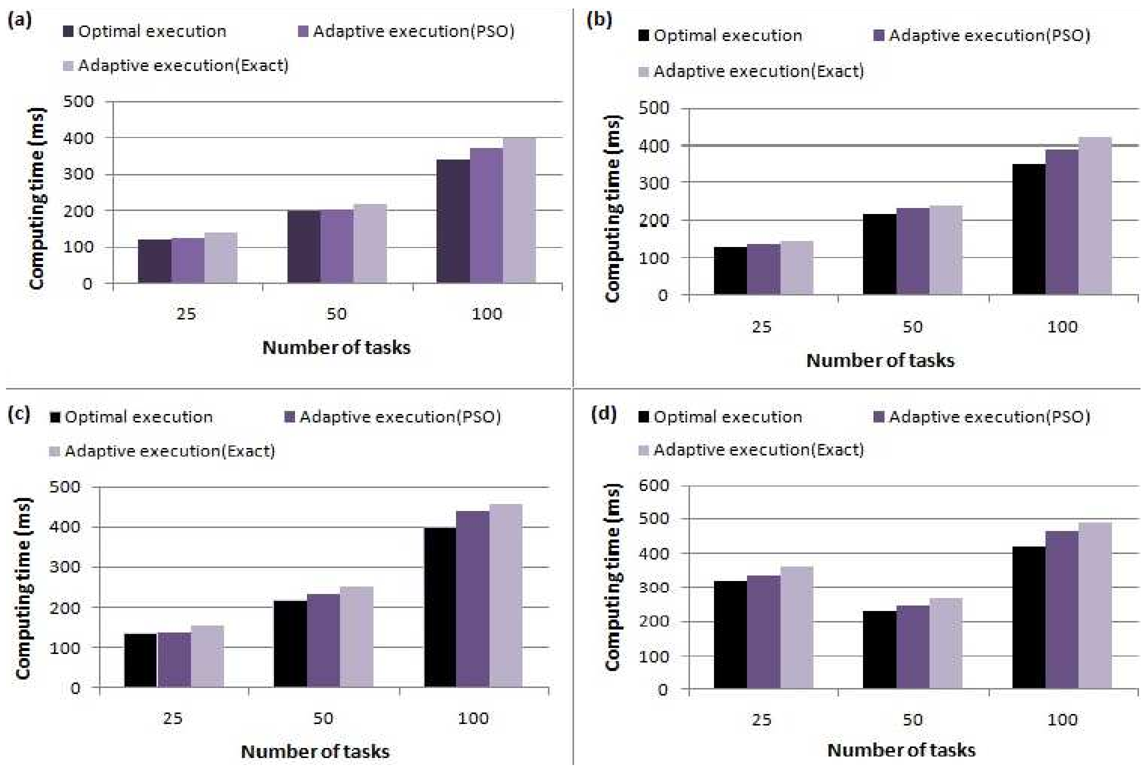


Fig. 6. The overhead of the adaptive execution using CloudSim4DWf simulator

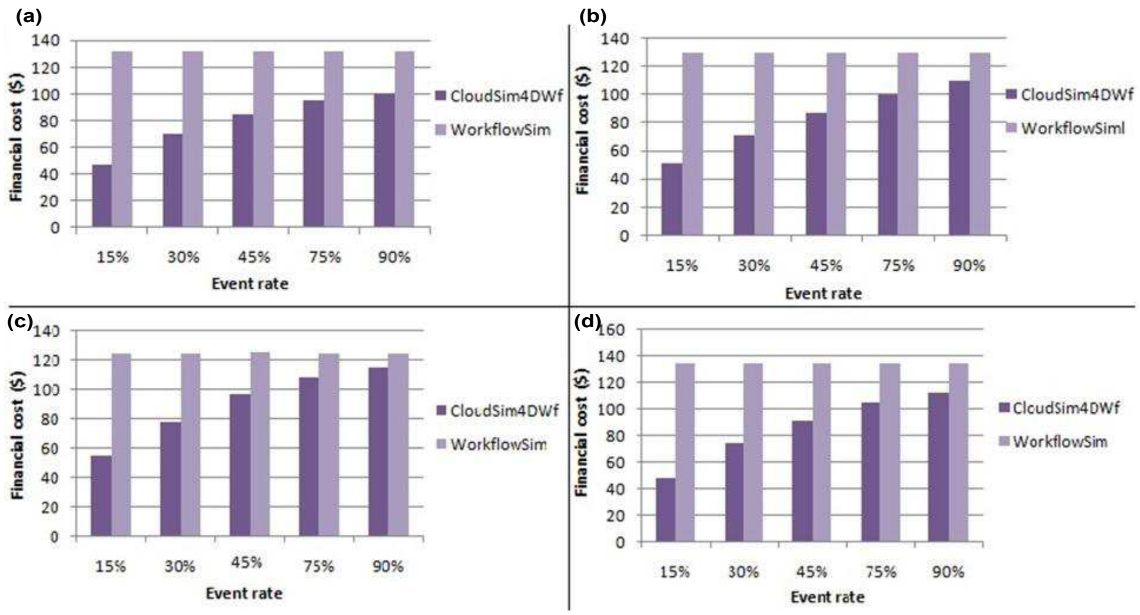


Fig. 7. Comparing the financial cost of our simulator with WorkflowSim

need to change the workflow instance at runtime. After that, it determines the resources appropriate to each task while meeting QoS requirements and the adaptation actions during execution. The different experiments that we have presented show the efficiency of our solution in terms of financial cost and the overhead caused by the remapping time.

As a future work, we plan to enhance our simulator by addressing other adaptation actions. Moreover, we aim to handle other workflow structures (such as LOOP and XOR). We plan also to perform a wide set of experiments and extend our resources model to consider federated cloud infrastructures that include multiple cloud providers and VM types.

References

[1] M. Bux and U. Leser, “DynamicCloudSim: Simulating heterogeneity in computational clouds,” *Future Generation Computer Systems*, vol. 46, pp. 85 – 99, 2015.

[2] R. Buyya and M. Murshed, “GridSim: a toolkit for the modeling and simulation of distributed resource management and scheduling for Grid computing,” *Concurrency and Computation: Practice and Experience*, vol. 14, no. 13-15, pp. 1175–1220, 2002.

[3] R. Calheiros, R. Ranjan, A. Beloglazov, C. De Rose, and R. Buyya, “CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms,” *Software: Practice and Experience*, vol. 41, no. 1, pp. 23–50, Jan. 2011.

[4] R. Calheiros, R. Ranjan, C. D. Rose, and R. Buyya, “Cloudsim: A novel framework for modeling and simulation of cloud computing infrastructures and services,” *CoRR*, vol. abs/0903.2525, 2009.

[5] W. Chen and E. Deelman, “WorkflowSim: A Toolkit for Simulating Scientific Workflows in Distributed Environments,” in *International Conference on E-Science*. IEEE, 2012, pp. 1–8.

[6] E. Deelman, D. Gannon, M. Shields, and I. Taylor, “Workflows and e-Science: An overview of workflow system features and capabilities,” *Future Generation Computer Systems*, vol. 25, no. 5, pp. 528–540, 2009.

[7] E. Deelman, G. Singh, M. Su, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, G. Berriman, J. Good, A. Laity, J. Jacob, and D. Katz, “Pegasus: A Framework for Mapping Complex Scientific Workflows Onto Distributed Systems,” *Scientific programming journal*, vol. 13, no. 3, pp. 219–237, 2005.

[8] F. Fakhfakh, H. Hadj-Kacem, and A. Hadj-Kacem, “Simulation tools for cloud computing: A survey and comparative study,” in *16th IEEE/ACIS International Conference on Computer and Information Science (ICIS)*, 2017.

[9] S. K. Garg and R. Buyya, “NetworkCloudSim: Modelling Parallel Applications in Cloud Simulations,” in *International Conference on Utility and Cloud Computing*, 2011, pp. 105–113.

[10] G. Juve, A. Chervenak, E. Deelman, S. Bharathi, G. Mehta, and K. Vahi, “Characterizing and profiling scientific workflows,” *Future Generation Computer Systems*, vol. 29, no. 3, pp. 682 – 692, 2013.

[11] K. Képes, U. Breitenbücher, S. Gómez Sáez, J. Guth, F. Leymann, and M. Wieland, *Situation-Aware Execution and Dynamic Adaptation of Traditional Workflow Models*. Cham: Springer International Publishing, 2016, pp. 69–83.

[12] H. Kim, Y. el Khamra, S. Jha, and M. Parashar, “Exploring application and infrastructure adaptation on hybrid grid-cloud infrastructure,” in *High Performance Distributed Computing*. New York, USA: ACM, 2010, pp. 402–412.

[13] D. Kliazovich, P. Bouvry, and S. Khan, “GreenCloud: a packet-level simulator of energy-aware cloud computing data centers,” *The Journal of Supercomputing*, vol. 62, no. 3, pp. 1263–1283, 2012.

[14] A. Kohne, M. Spohr, L. Nagel, and O. Spinczyk, “FederatedCloudSim: a SLA-aware federated cloud simulation framework,” in *Workshop on CrossCloud Systems*. ACM, 2014, pp. 1–5.

[15] S. Lim, B. Sharma, G. Nam, E. Kim, and C. Das, “MDCSim: A multi-tier data center simulation, platform,” in *International Conference on Cluster Computing*. IEEE, 2009, pp. 1–9.

[16] A. Núñez, J. Vázquez-Poletti, A. Caminero, G. Castañé, J. Carretero, and I. Llorente, “iCanCloud: A Flexible and Scalable Cloud Infrastructure Simulator,” *J. Grid Comput.*, vol. 10, no. 1, pp. 185–209, 2012.

[17] A. Salman, I. Ahmad, and S. Al-Madani, “Particle swarm optimization for task assignment problem,” *Microprocessors and Microsystems*, vol. 26, no. 8, pp. 363 – 371, 2002.

[18] S. Yeo and H. Lee, “Using Mathematical Modeling in Provisioning a Heterogeneous Cloud Computing Environment,” *Computer*, vol. 44, no. 8, pp. 55–62, August 2011.

[19] J. Yu, Q. Sheng, J. Y. Swee, J. Han, C. Liu, and T. Noor, “Model-driven development of adaptive web service processes with aspects and rules,” *Journal of Computer and System Sciences*, vol. 81, no. 3, pp. 533–552, 2015.

[20] A. Zhou, S. Wang, C. Yang, L. Sun, Q. Sun, and F. Yang, “FTCloudSim: support for cloud service reliability enhancement simulation,” *International Journal of Web and Grid Services*, vol. 11, no. 4, pp. 347–361, 2015.

[21] A. Zhou, S. Wang, Q. Sun, H. Zou, and F. Yang, “FTCloudSim: a simulation tool for cloud service reliability enhancement mechanisms,” in *Proceedings Demo & Poster Track of ACM/IFIP/USENIX International Middleware Conference*. Beijing, China: ACM, December 9-13 2013, pp. 1–2.