Resources allocation and scheduling approaches for business process applications in Cloud contexts

Kahina Bessai, Samir Youcef, Ammar Oulamara, Claude Godart, Selmin Nurcan

Resources allocation and scheduling approaches for business process applications in Cloud contexts

Kahina Bessai*, Samir Youcef**, Ammar Oulamara**, Claude Godart** and Selmin Nurcan*

*University of Paris1-Panthéon- Sorbonne
Centre de Recherche en Informatique
90, rue de Tolbiac 75634 Paris, France
kahina.bessai@gmail.com, nurcan@univ-paris1.fr

**University of Lorraine
LORIA-INRIA-UMR 7503
BP 239, Vandoeuvre-les-Nancy, France
{name}@loria.fr

***Abstract*—**

Resources allocation and scheduling has been recognised as an important topic for business process execution. However, despite the proven benefits of using Cloud to run business process, users lack guidance for choosing between multiple offering while taking into account several objectives which are often conflicting. Moreover, when running business processes it is difficult to automate all tasks. In this paper, we propose three complementary approaches for Cloud computing platform. On the other side, elastic computing, such as Amazon EC2, allows users to allocate and release compute resources (virtual machines) on-demand and pay only for what they use. Therefore, it is reasonable to assume that the number of virtual machine is infinite while the number of human resources is finite. This feature of Clouds has been called "illusion of infinite resources". In this paper, we design an allocation strategy for Cloud computing platform taking into account the above characteristics. More precisely, we propose three complementary bi-criterion approaches for scheduling business process on distributed Cloud resources.

I. Introduction

During the last decades software sales model dominated market. This model requires from customers managing continuously software deployment, including transitioning between different versions. Therefore, customers need technical expertise and expensive initial investment for acquiring and using them. Furthermore, they also need to pay for upgrades as annual maintenance fee. With the advent of Cloud computing paradigm, IT organizations need to think in terms of managing services rather than managing devices. Cloud service providers tent to offer services that can be classified in three categories [6][5]: i) Software as a Service (SaaS), ii) Platform as a Service (PaaS) and iii) Infrastructure as a Service (IaaS).

Therefore, the Cloud computing has quickly changed the way that compute resources can be used and allow users to access computing resources on the fly according to the application's need. For instance, to run any desired software, Amazon's EC2 [8] provides a Web service through which users can boot an Amazon Machine Image. However, despite the proven benefits of using Cloud to execute business processes, users lack guidance for choosing between different offering while taking into account several objectives often conflicting.

Moreover, most existing workflow matching and scheduling algorithms consider only an environment in which the number of resources is assumed to be bounded. However, in distributed systems such as Cloud computing this assumption is in opposition to the usefulness of such systems. Indeed, the "illusion of infinite resources" is the most important feature of Clouds [1][6], which means that users can request, and are likely to obtain, sufficient resources for their need at any time. Additionally to this characteristic, a Cloud computing environment can provide several advantages that are distinct from other computing environments: i) computing resources can be *elastically* scaled on demand (i.e. the number of resources used to execute a given workflow can be changed at runtime), ii) computing resources are exposed as services and thereby a standardization interface is provided. In this paper, we are interested to business processes where in opposition to scientific workflows it is difficult to automate all the tasks. Especially, certain tasks, that cannot be automated, require validation as subject to human intervention.

Furthermore, although there are efficient algorithms in the literature for workflow tasks allocation and scheduling for heterogeneous resources such as those proposed in grid computing context [19] [20] [21] [22], they do not consider the human dimension. In addition, they apply to a bounded number of resources. As a consequence, they are not appropriate for business processes matching and scheduling.

To overcome the limitations of existing approaches, we propose new approaches which are distinct from the related works as they mainly take into account Clouds elasticity feature and human resource dimension. Moreover, our approaches consider the overall completion time and the execution cost together.

To summarize, in this paper we make the following contributions:

1) We formalize a model for workflow tasks allocation and scheduling in Cloud environment taking into account the human dimension.
2) We propose a first approach for workflow tasks allocation and scheduling based on minimizing the execution cost incurred by using a set of resources.
3) We propose a second approach based on the overall completion time.
4) We present a third approach combining the above

objectives.

The remainder of the paper is organized as follows. In the next section, we introduce our workflow matching and scheduling model. Section III introduces the bi-criterion optimization approach. In Section IV, we present our matching and scheduling algorithms (the cost-driven approach, the time-driven approach and the cost-time-driven approach). Experimental results are presented in Section V. Section VI presents related works and compare them with our proposition. Finally, we draw conclusions and give some future works in Section VII.

II. Problem formulation and objective functions

Recall that the main scope of this paper is to deal with the workflow tasks allocation and scheduling problem in the Cloud environment. In the following, we start by refining the problem definitions and then present the cost and the time objective functions that we consider in this work. More precisely, we give a model comprising resources, role, tasks and business process applications definitions and the relationship between them.

A. Problem formulation

Before giving the problem statement, lets start with some definitions to facilitate the understanding of our approaches.

*Definition 1 (**Resource**):* A resource r represents an available unit that is required for executing a task. As mentioned earlier, a resource can be human resource or virtual machine. The set of available resources is denoted R

*Definition 2 (**Role**):* A role ro refers a class of resources that own the same capabilities. The set of available roles is denoted Ro.

*Definition 3 (**Task**):* A task t_i is a logical unit of work that is executed by a resource which can be human resource (if the task cannot be automated) or machine (if the task can be automated).

*Definition 4 (**Business process**):* A business process application is represented as a directed acyclic graph (DAG). Formally, a workflow application is a DAG represented by $G = (T, E)$, where:

1) $T = \{t_1, ..., t_n\}$ is a finite set of tasks.
2) E represents the set of directed edges. An edge (t_i, t_j) of graph G corresponds to the data dependencies between these tasks (the data generated by t_i is consumed by t_j).
3) Task t_i is called the immediate parent of t_j which is the immediate child task of t_i.

Note that the child task cannot be executed until all of its parents tasks are completed. In a given graph, a task without any precedents is called an input task, denoted t_{input} and a task without successors is called an exit task, denoted t_{exit}.

*Definition 5 (**Data**):* Let $Data$ be a $n \times n$ matrix of communication data, where $data[i, j]$ is the amount of data required to be transmitted from task t_i to task t_j.

Most of the workflow tasks allocation and scheduling algorithms require single input (t_{input}) and single exit (t_{exit}) task graphs. So, if there is more than one input (exit) task, they are connected to a zero-cost (-time) `pseudo-input (-exit)` task with zero communication cost and time, which does not affect the allocation and the schedule.

Moreover, a task is an indivisible unit of work and is non-preemptive. Figure 1 (left side) shows an example of workflow application defined by ten tasks, which are represented as nodes. The dependencies between tasks are represented as arrows. The initial task may have an input file denoted (`input.file`) and the exit task produces an output file (`output.file`).

To execute a given workflow application (DAG), a set of resources (human resources and virtual machines) can be used on-demand.

*Definition 6 (**Resource graph**):* The resources are represented as a directed graph denoted RG. Formally, a resources graph is represented by $RG = (R, V)$, where:

1) $R = \{VM_1, ..., VM_m, HR_1, ..., HR_{m'}\}$ is a finite set of virtual machines types and human resources.
2) V represents the set of directed edges. Each edge is denoted (VM_i, VM_j) corresponding to the link between these virtual machines.

However, unlike the number of human resources which is assumed to be finite, we consider that there is enough virtual machines for each type. Thus, a user can request and obtain sufficient virtual machines at any time. This assumption is reasonable in Cloud computing environment because it gives for user an "illusion of infinite resources". When there is no ambiguity, we omit the term type and use virtual machine instead virtual machine type.

*Definition 7 (**Bandwidth**):* Let B be a $m \times m$ matrix, in which $B[i, j]$ is the bandwidth between virtual machine types VM_i and VM_j, where $B[i, i] \longrightarrow \infty$ means that there is no transfer data.

Remark It is important to note that data are transmitted only through the used virtual machines. In others words, if a given task is performed by a human resource, it uses a virtual machine, once finished executing this task, to transmit the generated data to any other resources that will execute the next tasks.

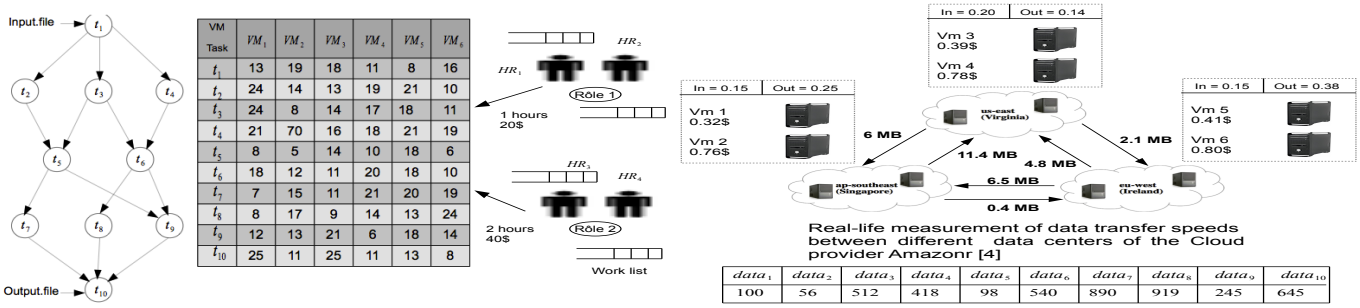
Figure 1 (right side) shows an example of resources graph with real-life measurement of data transfer speeds (bandwidth) between different data centers of the Cloud provider Amazon. To simplify the example, we assume that $data[i, j] = data[i]$.

*Definition 8 (**Transfer time**):* Let $r(t_j)$ denotes the resource (virtual machine or human resource) that executes task t_j. The transfer time $TT(r(t_i), r(t_j))$, which is for transferring data from task t_i (executed by $r(t_i)$) to task t_j (executed by $r(t_j)$) is defined by:

$$TT(r(t_i), r(t_j)) = \frac{data[i, j]}{B[r(t_i), r(t_j)]}$$

*Definition 9 (**Execution time matrix**):* Let ET be a $n \times m$ execution time matrix in which $ET(t_i, r_j)$ gives the execution time estimation to complete task t_i by resource r_j.

*Definition 10 (**Unit execution cost vector**):* Let UEC be a $(m + m')-$dimensional unit execution cost vector, where $UEC(r_j)$ represents the cost per time unit incurred by using the resource r_j. Let EC be a $n \times (m + m')$ execution cost matrix in which $EC(t_i, r_j)$ gives the execution cost to complete task t_i by resource r_j defined by:



VM / Task	VM_1	VM_2	VM_3	VM_4	VM_5	VM_6
t_1	13	19	18	11	8	16
t_2	24	14	13	19	21	10
t_3	24	8	14	17	18	11
t_4	21	70	16	18	21	19
t_5	8	5	14	10	18	6
t_6	18	12	11	20	18	10
t_7	7	15	11	21	20	19
t_8	8	17	9	14	13	24
t_9	12	13	21	6	18	14
t_{10}	25	11	25	11	13	8

$data_1$	$data_2$	$data_3$	$data_4$	$data_5$	$data_6$	$data_7$	$data_8$	$data_9$	$data_{10}$
100	56	512	418	98	540	890	919	245	645

Fig. 1. An illustrative business process application and a set of resources (virtual machines and human resources)

$$EC(t_i, r_j) = ET(t_i, r_j) \times UEC(r_j)$$

*Definition 11 (**Transfer cost**):* The data transfer cost $TC(r(t_i), r(t_j))$, which is the cost incurred due to the transfer of data from task t_i (executed by $r(t_i)$) to task t_j (executed by $r(t_j)$), is defined by:

$$TC(r(t_i), r(t_j)) = data[i, j] \times (C_{out}(r(t_i)) + C_{in}(r(t_j)))$$

where $C_{out}(r(t_i))$ and $C_{in}(r(t_j))$ represent respectively the cost of transferring data from $r(t_i)$ and the cost of receiving data on $r(t_j)$. The transferring data cost is determined by mutual agreement between the consumer and the provider in the SLA (e.g Amazon CloudFront).

B. Objective functions

In the following, we give the formal definition of the considered objective functions in this paper (i.e. the overall execution time and the cost incurred using a set of resources). Before giving the objective functions considered in our study, we start by the the following definition.

*Definition 12 (**Human ressource availability**):* The availability of resource r_j, denoted $avail[j]$, is the earliest time at which the resource r_j is ready for task execution.

1) a. Time objective function: Let EST (earliest start time) and EFT (earliest finish time) attributes that characterize the set of resources (virtual machine types and human resources). These attributes are derived from a given partial allocation and scheduling (i.e. a task t_i is assigned to virtual machine $VM(t_i)$ or human resource $HM(t_i)$). The partial schedule refers to the fact that for each task the earliest start time and the earliest finish time values are obtained using only the tasks that must be performed before it. $EST(t_i, r_j)$ and $EFT(t_i, r_j)$ are the earliest execution start time and the earliest execution finish time of task t_i on resource r_j, respectively. For the input task t_{input}, the earliest execution start time and the earliest execution finish time are given by Equation 1 and Equation 2, respectively:

$$EST(t_{input}, r_j) = 0 \tag{1}$$

$$EFT(t_{input}, r_j) = ET(t_i, r_j) \tag{2}$$

For the other tasks in the graph, the EST and the EFT values are computed recursively, starting from the initial task, as shown in Equation 3 and Equation 4. In order to compute the EFT of a task t_j, all immediate predecessor tasks of t_j must have been assigned and scheduled with the consideration of the transfer time.

$$EFT(t_i, r_j) = EST(t_i, r_j) + ET(t_i, r_j) \tag{3}$$

$$EST(t_i, r_j) = \max\left\{avail[j], \max_{t_p \in pred(t_j)}[AFT(t_p) + TT(r(t_p), r(t_i))]\right\} \tag{4}$$

where $pred(t_i)$ is the set of immediate predecessors of task t_i and $avail[j]$ is the earliest time at which resource r_j is ready for task execution. As the number of virtual machines is assumed to be infinite, then $avail[j] = 0$ where r_j is a virtual machine. In other words, if task t_i is performed by resource r_j which is a virtual machine then $EST(t_i, r_j)$ is computed as follows:

$$EST(t_i, r_j) = \max_{t_p \in pred(t_j)}\{AFT(t_p) + TT(r(t_p), r(t_i))\} \tag{5}$$

However, as the number of human resources is assumed to be finite, then if task is performed by human resource the earliest time at which this resource is ready should be taking into account (i.e. $avail[j]$). Therefore, if t_i is the last assigned task on resource r_j, then $avail[j]$ is the time that resource r_j completed the execution of the task t_i and it is ready to execute another task. The inner max block in the EST equation returns the ready time, i.e. the time when all required data by t_i has arrived at resource r_j.

After a task t_i is scheduled on a resource r_j, the earliest start time and the earliest finish time of t_i on resource r_j is equal to the actual start time, denoted $AST(t_i)$, and the actual finish time, denoted $AFT(t_i)$, of task t_i, respectively.

After all tasks in a graph are scheduled, the schedule length (i.e., the overall completion time) will be the actual finish time of the exit task (i.e. $AFT(t_{exit})$). The schedule length, also called *makespan*, is defined as:

$$makespan = AFT(t_{exit}) \tag{6}$$

Therefore, the *time objective function* is to determinate the assignment of tasks of a given workflow application to resources (virtual machines and human resources) such that its schedule length is minimized.

Predictive heuristic for human resource availabilities The actual finish time as computed previously does not take into account the fact that human resources can perform other tasks that do not belong to the same process. More precisely, the work list of a given human resource may contain work

items of different processes. The realism of this assumption can be disputed as human resources are "shared" by more than one process. Thus, it is might be desirable to design a procedure which tries to predict the humane resource availabilities. Concretely, we propose to estimate the availability values of human resources taking into account the previous observations. Let $availability[j]_{k+1}$ the estimation of resource j availability and t_i the next task that can be performed by this resource. Instead of simply adapting to the computed availability using equation 4, one can try to forecast and estimate what $availability[j]_{k+1}$will be using the historical data. In our experiment results we used a double exponential smoothing to estimate the availability of a given resource. Formally, the model is represented by the following equation:

$$Avail[j]_{k+1} = (1-\alpha)\sum_{m=0}^{k}(1-\alpha)avail[j]_{k-m}\ ,\ 0<\alpha<1 \quad (7)$$

Intuitively, Equation 7 produces a prediction of resource availabilities that weights historical data based on exponential decay according to the time distance in the past. The parameters α is computed from the data using auto-correlation analysis.

2) b. Cost objective function: In the following, we focus on the cost objective function which is the total expense for workflow tasks execution including (i) the tasks execution cost and (ii) the data transfer cost between the used virtual machines.

The cost function is a structure independent criterion defined as the sum of the costs of executing all workflow tasks, given by:

$$cost = \sum_{j=1}^{n}\left\{EC(r(t_j)) + \sum_{p\in pred(t_j)} TC(r(t_p), r(t_j))\right\} \quad (8)$$

Thus, the *cost objective function* is to determinate the assignment of tasks of a given workflow application such that its overall execution cost is minimized.

Problem statement Suppose a business process together with its given task set T, resource set R, role set Ro, a mapping from resources to role, i.e. $R \rightarrow Ro$, and from role to tasks, i.e. $Ro \rightarrow T$. The objective is to find a set of resources to execute workflow applications while minimizing the overall execution time and the cost function simultaneously. Formally, this problem is defined as follows:

$\min\{AFT(t_{exit})\}$ and
$\min\sum_{j=1}^{n}\left\{EC(r(t_j)) + \sum_{p\in pred(t_j)} TC(r(t_j), r(t_p))\right\}$

$$\begin{cases} respect\ the\ tasks-precedence\ requirement\ (DAG) \\ given\ a\ set\ of\ resources\ (R) \end{cases}$$

III. Bi-criteria approach

The optimization problem defined above can be approached in several ways. Indeed, to deal with bi-criterion allocation and scheduling problems mainly three streams are considered [14] [15] [16]: i) aggregation approach, ii) ϵ-approach and iii) Pareto approach. By studying the possible transformation of the addressed problem into a mono-criteria one, we remarked that no criterion is dominant. Therefore, the Pareto approach seems us the most appropriate for addressing our problem. A multi-objective problem can be defined as:

$$\min f(x) = (f_1(x), ..., f_n(x))$$

where $x \in X$; X is a set of feasible solutions (or decision space). A feasible solution $x \in X$ is a Pareto solution (or efficient solution) if there does not exist any $x' \in X$ such that:

$$\forall i \in \{1, ..., n\}, f_i(x') \leq f_i(x)\ \wedge\ \exists j \in \{1, ..., n\}, f_j(x') < f_j(x)$$

Figure 2 shows solutions for the minimization problem of two conflicting objectives $f_1(x)$ and $f_2(x)$. The solution x_4 dominates x_1, because both objective values of x_4 are lower than those of x_1 (i.e. $f_1(x_4) < f_1(x_1)$ and $f_2(x_4) < f_2(x_1)$). However, x_2 does not dominate x_4, since $f_1(x_2) > f_1(x_4)$. We say that x_4 and x_2 are non-dominated solutions (Pareto solutions). The set of non-dominated solutions forms a non-dominated front, denoted $PS \subseteq X$ (see Figure 2).

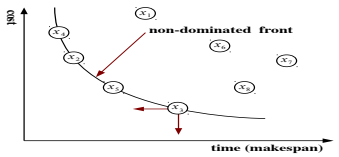


Fig. 2. Four non-dominated solutionand four dominated solutions

IV. Approaches for tasks allocation and scheduling in Cloud contexts

In the following, we detail the three proposed complementary approaches. The first one is based on the minimization of the overall execution time, while the second one aims to minimize the cost incurred using a set of resources. The third approach allows to select only the Pareto solutions obtained by the two approaches.

A. The cost-based approach

In the cost-based approach, we focus only on minimizing the execution and communication costs of using a set of resources incurred by the execution of a given process. However, for each obtained feasible solution by using an allocation strategy the overall corresponding completion time is computed. Recall that the objective is to assign tasks to resources respecting the precedence constraints. The cost-based approach is an application allocation and scheduling algorithm for an unbounded number of virtual machines and a bounded number of human resources. As mentioned previously, users can request and obtain sufficient virtual machine at any time, while a human resource cannot execute more than one task at any time (i.e. human resources cannot execute tasks in parallel). The approach proposed has three major phases, namely: i) tasks sorting phase, ii) resource allocation phase and iii) Pareto selection phase. An overview of the cost-based approach is shown by Algorithm 1.

1) Tasks sorting phase: In order to group the tasks that are independent of each other, the given business process (DAG) is traversed in a top-down fashion to sort tasks at each level. As a result, tasks belonging to the same level do not exchange data

and can be executed in parallel (because they are not related by precedence constraints). Given a DAG $G = (T, E)$, the level 1 and the last level contains respectively the input t_{input} task and the exist t_{exit} task. Level k, denoted l_k, consists of all tasks t_j such that for all edges (t_i, t_j), task t_i is in a level $k' < k$ and there exists at least one edge (t_i, t_j) such that t_i is in level $k-1$. Let L be the number of levels of a given process.

For instance, for the tasks graph given in Figure 1, there are five levels (i.e. $L = 5$): level l_1 consists of task t_1 (initial task), level l_2 consists of task t_2, t_3, t_4, level l_3 consists of tasks t_5, t_6, level l_4 consists of task t_7, t_8, t_9 and the level five l_5 contains task t_{10} (exit task). The line 2 of Algorithm 1 corresponds to the tasks sorting phase.

2) Resource allocation phase: In this phase the selection of an "optimal" resources for each task is decided. In other words, the resource which gives minimum execution and communication costs for a task is selected and the task is assigned to that resources. More precisely, given the labeling of tasks in the graph levels, the allocation process explores the graph by starting the allocation tasks of level k, where the value of k is given by the following strategies: i) top-down, iii) bottom-up and iii) mixed exploration and allocation strategy.

3) The top-down: strategy consists of starting by the allocation of the initial task (level l_1) to the resource which gives minimum execution cost. After this assignment, the graph is traversed in a top-down fashion from level 2 to level L. At level k, the task is assigned to the resource which gives minimum of execution and communication costs as follows:

$\forall t_i \in l_k, r(t_i) = r_s$ such that:
$EC(t_i, r_s) + \sum_{t_h \in pred(t_i)} TC(r(t_h), r(t_i)) =$

$$\min_{r_j} \left\{ EC(t_i, r_j) + \sum_{t_h \in pred(t_i)} TC(r(t_h), r(t_i)) \right\} \quad (9)$$

where $pred(t_i)$ is the set of immediate predecessors of task t_i.

The lines 3 (with $k = 1$) to 10 of Algorithm 1 corresponds to the top-dow strategy.

4) The bottom-up: strategy consists on starting by the allocation of the finish task (the last level). After this allocation, the graph is traversed in a bottom-up fashion from level $L-1$ to level 1. At level k, the task is assigned to the resource which gives minimum of execution and communication costs as follows:

$\forall t_i \in l_k, r(t_i) = r_s$ such that:
$EC(t_i, r_s) + \sum_{t_h \in succ(t_i)} TC(r(t_i), r(t_h)) =$

$$\min_{r_j} \left\{ EC(t_i, r_j) + \sum_{t_h \in succ(t_i)} TC\left(r(t_i), r(t_h)\right) \right\} \quad (10)$$

where $succ(t_i)$ is the set of immediate successors of task t_i.

The lines 11 (with $k = L$) to 15 of Algorithm 1 corresponds to the bottom-up strategy.

5) The mixed: strategy starts by assigning the tasks belonging to the intermediate level, i.e. $k \in \{2, ..., L-1\}$. Given starting level k, therefore the assignment of the tasks belonging to this level is only based on the execution cost $EC(t_i, r_j)$, given by the following equation:

$\forall t_i \in l_k, r(t_i) = r_k$ such that:

$$EC(t_i, r_k) = \min_{r_j} EC(t_i, r_j) \quad (11)$$

For the tasks belonging to the level $k' < k$ and $k' > k$, the assignment of a task t_i is impacted by the previous assignments by taking into account the both costs (execution and communication costs), using bottomp-up and top-down strategy respectively. Equation 9 and 10 are recursively applied until all the tasks are assigned using the mixed strategy for all $k \in \{2, ..., L-1\}$. In the case of $k \in \{1, L\}$, we use the top-down and bottom-up strategy respectively. Note that, equations 9 and 10 have respectively one variable ($r(t_i)$) because $r(t_h)$ is computed in the previous iteration.

The lines 4-17 ($k \in \{2, ..., L-1\}$) of Algorithm 1 corresponds to the mixed strategy.

Hence, for the mixed strategy we obtain $L-2$ solutions (each of them consists on the assignment of all tasks) and one solution respectively for top-down and bottom-up strategy.

6) Pareto selection phase: Recall that at the end of the previous phase a L (the number of levels of a given process) assignment of all tasks is obtained. In this phase, we first compute the overall completion time (using Equation 2, 3 and 4) corresponding to each assignment and then only non-dominated solutions are maintained. The lines 18 and 19 of Algorithm 1 corresponds to the Pareto selection phase.

Algorithm 1 Cost-based approach

1: read the DAG, the RG and associated attributes values;
2: sort tasks at each level by traversing the DAG in a top-down fashion; // let L be the set of levels and l_k the tasks // belonging to the level k
3: $k \leftarrow 1$; // first level
4: **while** ($k \leq L$) **do**
5: for all tasks $t_i \in l_k$, compute $r(t_i)$ using equation 11
// assign task t_i to the resource $r(t_i)$
// that minimizes the execution cost
$mincost[k, t_i] \leftarrow r(t_i)$; // $mincost$ is a $L \times m$ matrix
// where line k corresponds to the assignment of all tasks
// obtained starting by the tasks assignement belonging to
// this level
6: $h \leftarrow k + 1$; // compute $r(t_i)$ for all tasks that belong to levels $h > k$
7: **while** ($h \leq L$) **do**
8: for all tasks $t_i \in l_h$, compute $r(t_i)$ using equation 9
$mincost[k, t_i] \leftarrow r(t_i)$;
9: $h \leftarrow h + 1$
10: **end while**
11: $h \leftarrow k - 1$; // compute $r(t_i)$ for all tasks that belong // to levels $h < k$
12: **while** ($h \geq 1$) **do**
13: for all tasks $t_i \in l_h$, compute $r(t_i)$ using equation 10
$mincost[k, t_i] \leftarrow r(t_i)$;
14: $h \leftarrow h - 1$
15: **end while**
16: $k \leftarrow k + 1$
17: **endwhile**
18: for each assignment, compute $AFT(t_{exit})$ using equations 2, 3 and 4;
19: select the Pareto solutions among L solutions;

B. The time-based approach

While the previous approach is based on minimizing the *cost* function, the time-based approach, detailed in the following, is based on the *makespan* criterion. More precisely, the time-based approach attempts to minimize the overall completion time (i.e. execution and communication time). As the cost-based approach, the time-based approach is an application matching and scheduling algorithm for an "unbounded" number of virtual machines and bounded number of human resources, which has three major phases, namely: a tasks sorting phase i), ii) an allocation phase and iii) Pareto selection phase. This approach contains the same steps as the previous approach. The Equations 11, 9 and 10 are replaced by Equations 12, 13 and 14 respectively.

1) Tasks sorting phase: This phase is the same as for the cost-based algorithm. Recall that this phase allows to group the workflow application tasks that are independent of each other.

2) Resource allocation phase: The cost-based and the time-based approaches differ mainly at resource allocation phase. Indeed, the first one approach focus on minimizing the *cost* function while the second approach attempts to minimize the *time* function. The objective of the resource allocation phase of the time-based approach is to choose the resource that minimizes the actual finish time of each task and the task is assigned to that resource. The three allocation strategies, detailed above, are also applied to explore the given graph (DAG). Therefore, the obtained solutions number is equal to the number of graph levels (i.e. L). So, If the resource allocation phase starts at the level k, therefore the assignment of the tasks belonging to this level is only based on the execution time $ET(t_i, r_j)$, given by the following equation:

$\forall t_i \in l, r(t_i) = r_k$ such that:

$$ET(t_i, r_k) = \min_{r_j} ET(t_i, r_j) \tag{12}$$

For the tasks belonging to the level l_{k+1} and l_{k-1}, the allocation decision of a task t_i are recursively defined by respectively the following equations until all the tasks are assigned:

$\forall t_i \in l_{k+1}, r(t_i) = r_k$ such that:

$ET(t_i, r_k) + \sum_{t_h \in pred(t_i)} TT(r(t_h), r(t_i)) =$

$$\min_{r_j} \left\{ ET(t_i, r_j) + \sum_{t_h \in pred(t_i)} TT(r(t_h), r(t_i)) \right\} \tag{13}$$

$\forall t_i \in l_{k-1}, r(t_i) = r_k$ such that:

$ET(t_i, r_k) + \sum_{t_h \in succ(t_i)} TC(r(t_i), r(t_h)) =$

$$\min_{r_j} \left\{ ET(t_i, r_j) + \sum_{t_h \in succ(t_i)} TC\left(r(t_i), r(t_h)\right) \right\} \tag{14}$$

3) Pareto selection phase: Recall that at the end of the previous phase L assignment of all tasks is obtained. In this phase, we first compute the overall execution cost (using Equation 8) corresponding to each assignment and then only non-dominated solutions are maintained.

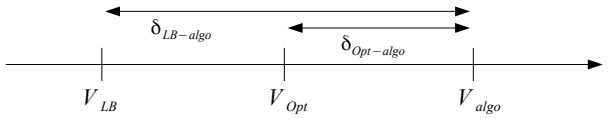


Fig. 3. Gap between the obtained solutions, the optimal and the lower solutions

C. Cost-time-based approach

The cost-time-based approach objective is to offer a guidance for choosing between different offering in order to complete the workflow application considering the two criterion together. It is composed by two phases. The first one consists to execute both algorithms (cost-based and time-based algorithms). The second one allows to select only the non-dominated solutions.

V. Experimental results and discussion

In order to assess the quality of the obtained solutions by our algorithms, we need to compare the obtained solutions with an optimal solution of the problem. But as the problem is NP-hard, the optimal solution cannot be obtained in polynomial time, thus a relaxation of the original problem is needed to obtain a lower bound of the optimal solution in polynomial time. Then instead to compare our solutions with optimal solution of the original problem, we compare it with the lower bound. As showed in Figure 3, $\delta_{LB-algo}$ is the gap between the solution of our algorithm (V_{algo}) and the lower bound (VLB), whereas $\delta_{Opt-algo}$ is the effective gap between the optimal solution (V_{opt}) and the solution obtained by our algorithm.

A. Lower bounds

For time criterion, a lower bound LB_t is obtained by the execution of the t_{exit} task in the critical path of the relaxed problem for the workflow $G = (T, E)$, in which each task t_j is executed on the resource $r(t_j)$ with the minimal execution time, i.e., $ET(t_j, r_j) = \min_{r_k} ET(t_j, r_k)$ and each edge (t_i, t_j) of G is valued with a lower bound of the transfer time between t_i and t_j. The transfer time $TT(i, j)$ is defined by the following formula:

$$TT(t_i, t_j) = \min \begin{cases} ET(t_j, r(t_i)) - ET(t_j, r(t_j)) & (1) \\ ET(t_i, r(t_j)) - ET(t_i, r(t_i)) & (2) \\ ET(t_i, t_j, r_k) & (3) \end{cases}$$

where $ET(t_i, t_j, r_k) = \min_{r_l}\{ET(t_i, r_l(t_j)) - ET(t_i, r(t_i)) + ET(t_j, r_l(t_j)) - ET(t_j, r(t_j))\}$. In the formula of $TT(t_i, t_j)$, the first term (1) represents the additional execution time of task t_i if it is executed on resource $r(t_j)$ instead of $r(t_i)$, the second term (2) represents the additional execution time of task t_j if it is executed on resource $r(t_i)$ instead of $r(t_j)$, and the last term (3) represents the additional execution times of tasks t_i and t_j if they are executed on resource $r_k(t_i)$ instead of $r(t_i)$, and on resource $r_k(t_j)$ instead of $Vr(t_j)$, respectively.

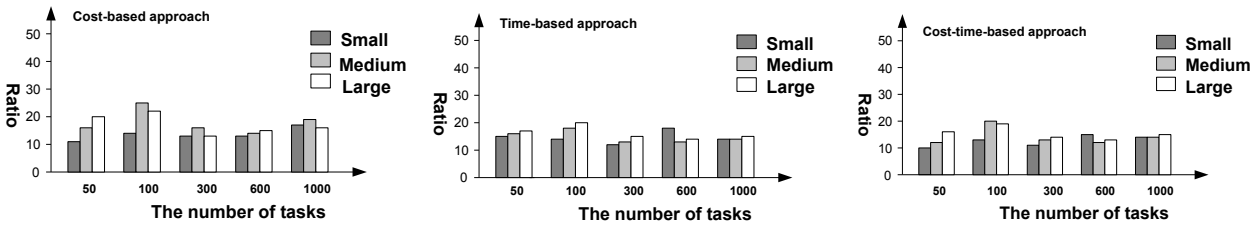


Fig. 4. Scheduling 1000 randomly generated workflows versus the ratio of the obtained Pareto solutions for each proposed approach

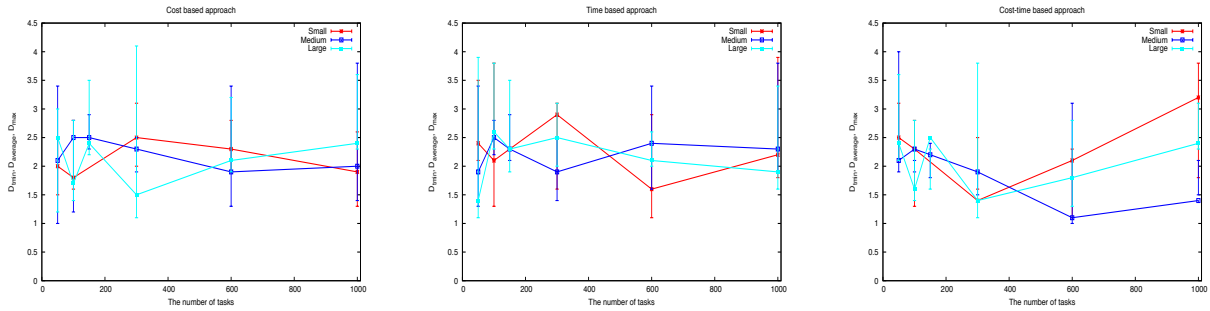


Fig. 5. Scheduling 1000 randomly generated workflows versus the ratio D_{min}, $D_{average}$ and D_{max} of the obtained Pareto solutions

B. Data setting

In our experiment, we simulate a Cloud computing environment in which five families of instances are randomly generated. A family is associated on tasks $n \in \{50, 100, 300, 600, 1000\}$. Each family of instances is subdivided into three series S, M and L with small, medium and large density of precedence constraints, respectively. For each pair (t_i, t_j) of tasks it is decided with a certain probability p that t_i precedes t_j, such that G is acyclic. For the series S, M, and L the corresponding probabilities are $p_S = 0.2$, $p_M = 0.4$ and $p_L = 0.6$. We consider the number of virtual machines types m as a function of the number of tasks, i.e., $m = n/10$. Also, the number of human resources is $m' = n/10$. As mentioned previously, for the real resource availability we use exponential smoothing. We assume that each task can be executed on all virtual machines types. For all families of instances, processing times of tasks on virtual machines are generated according to an uniform distribution, varying in $[1, 10]$. The unit execution cost on each virtual machine type is uniformly generated in interval $[0.1, 0.9]$. The bandwidth matrix between virtual machines is uniformly generated, in the interval $[1, 9]$ and the transfer data matrix D is also uniformly generated in interval $[10, 90]$. Finally the transfer C_{out} and the receiving cost C_{in} of virtual machines are uniformly generated in the interval $[1, 9]$. Each series consists of 1000 instances.

C. Summary of results and discussion

The obtained Pareto solutions divided by the number of the overall obtained solutions is considered as the ratio of each approach. Figure's 4 histograms indicate two things. First, in most cases the ratio increases with the density of precedence constraints (i.e p), which is mainly due to the importance of time and cost communications. So, It is interesting to consider not all levels of the graph, but only those where the obtained solution is very likely a Pareto solution. Second, the ratio of Pareto solutions obtained by the cost-time based approach is almost equal to the sum of the ratios of the cost-based and the time-based approach divided by two, which means that no criterion dominates the other one. Therefore, the Pareto approach is the most appropriate one to deal with the allocation and scheduling workflow tasks in Cloud computing environment.

The result of the quality of the obtained solutions by our approaches is given in Figure 5, plotting the minimum D_{min}, the average $D_{average}$ and the maximum D_{max} performance criterion. The D_{min}, $D_{average}$ and D_{max} are obtained by dividing the minimum value, the average value and the maximum value among all obtained Pareto solutions and the lower bound, respectively, by considering the two criteria (i.e. cost and time). These results show that in average the considered performance criterion do not exceed 2.9, which means that the obtained Pareto solutions are less than to 2.9 times the lower bound. Note that this observation is the same in three proposed approaches.

VI. Related work and discussion

Due to its importance on performance, the workflow tasks allocation and scheduling problem on heterogeneous computing environment such as grid has been intensively explored. Beside, the tasks allocation to compute resources is an NP-complete problem in the general form [13]. So, past works have proposed heuristic driven approaches for scheduling workflow

applications [17] [18]. These heuristics cannot be applied in Clouds computing environments because they require a bounded number of resources.

Therefore, allocation and scheduling workflow tasks in Cloud has gained popularity in recent times and few algorithms are proposed to deal with this problem [19] [20] [21] [22]. In [19], the authors developed a model that uses particle swarm optimization (PSO) for task-resource mapping to minimize the overall cost of execution such that it completes within deadline that user specifies. To tackle the problem of choosing resource among different cloud providers, a binary integer program is proposed in [19], where the objective is to minimize the total infrastructure capacity under budget and load balancing constraints. In [20], the authors presented a model for formulation of the generalized federated placement problem and application of this problem to load balancing and consolidation within a cloud, where one cloud can subcontract workloads to partnering clouds to meet peaks in demand. They used an Integer Program formulation of the placement program and provide a 2-approximation algorithm. In [21], the authors proposed a binary integer program formulation for cost-optimal scheduling in hybrid IaaS clouds for deadline constrained workloads. In [22], the authors proposed a set of heuristics to cost-efficiently schedule deadline-constrained computational applications on both public cloud providers and private infrastructure. Although, most of these studies consider unbounded number of resources, however, they convert the initial problem (bi-cretiria) to the $\epsilon-$ constraints problem.

To overcome these limitations, we propose new approaches which are distinct from the related work as they take into account Clouds elasticity feature, which means that users can rent and release resources according to the need of its applications. Moreover, our approaches consider the overall completion time and the execution cost together, where the problem of tasks allocation and scheduling problem has not been converted neither to mono-criterion nor to $\epsilon-$constraints problem. Therefore, the matching and scheduling problem described here does not appear to have been studied before. The work presented in this paper is an extension of our previous works [2] [3].

VII. Conclusion

This paper deals with the allocation and scheduling workflow tasks in Cloud contexts. More precisely, as the problem is computational hard we have proposed three approaches based respectively on the overall execution time, on the cost incurred using a set of resources (i.e. virtual machines and human resources) and on the both criterion. The obtained results are very encouraging but one the same time open new and interesting questions. We focus on several perspectives. First, we interested in applying the proposed approaches on real-world applications. Then, we want to compare the obtained results with the obtained ones in the case of average-analysis. Finally, we interested to extend the proposed approach to take into account more complex workflow patterns.

References

[1] *Workflow Management Coalition: Terminology & Glossary,* WfMC-TC-1011, 1999.

[2] K. Bessai, S. Youcef, A. Oulamara, C. Godart, S. Nurcan, *Bi-criteria workflow tasks allocation and scheduling in Cloud computing environments,* Cloud Computing 2012, pp. 638?645.

[3] K. Bessai, S. Youcef, A. Oulamara, C. Godart, S. Nurcan, *Multi-objective Resources Allocation Approaches for Workflow Applications in Cloud Environments,* OTM 2012 Workshops, LNCS 7567, pp. 654-657, 2012.

[4] R. Buyya, S. Pandey and C. Vecchiola,*Cloudbus toolkit for market-oriented cloud computing*, In CloudCom'09: Proceedings of the 1st International Conference on Cloud Computing, volume 5931 of LNCS, pp. 24-44, Springer, Germany, December 2009.

[5] P. Mell, T. Grance, *The NIST Definition of Cloud Computing*, (Draft)-Recommendations of the National Institute of Standards and Technology. Special publication 800-145 (draft), Gaithersburg (MD).

[6] R. Buyya, S. Pandey and C. Vecchiola, *Cloudbus toolkit for market-oriented cloud computing*, In CloudCom'09: Proceedings of the 1st International Conference on Cloud Computing, volume 5931 of LNCS, pp. 24-44, Springer, Germany, December 2009.

[7] G. Juve and E. Deelman, *Scientific Workflows in the Cloud, in Grids, Clouds and Virtualization*, M. Cafaro Eds. Springer, pp. 71-91, 2010.

[8] G. Juve, E. Deelman, K. Vahi,, G. Mehta, B. Berriman, B. P. Ber-man and P. Maechling, *Scientific workflow applications on Amazon EC2*. In Cloud Computing Workshop in Conjunction with e-Science, Oxford, UK, 2009, IEEE.

[9] A. V. Mladen, *Cloud Computing - Issues, Research and Implementations*, in Journal of Computing and Information Technology, Volume 16, Issue 4, 2008, pp. 235-246.

[10] 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, Volume 25, Issue 5, pp. 528-540, 2009.

[11] H. Topcuoglu, S. Hariri, M. Y . Wu, *Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing*, IEEE Transactions on Parallel and Distributed Systems, Volume 13, Issue 3, 2002.

[12] J. D. Ullman, *NP-Complete Scheduling Problems*, J. Computer and Systems Sciences, Volume 10, pp.3 84-393, 1975.

[13] M. R. Gary and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman and Co., 1979.

[14] H. Hoogeveen,*Multicriteria scheduling*, In European Journal of Operational Research 167, pp. 592-623, 2005.

[15] M. Wieczorek, S. Podlipnig, R. Prodan, T. Fahringer, *Bi-criteria Scheduling of Scientific Workfows for the Grid*, Eighth IEEE International Symposium on Cluster Computing and the Grid, pp 9-16, 2008.

[16] V. T'kindt, J. C. Billaut, *Multicriteria Scheduling. theory, models and algorithms*, Springer, 2nd Edition, 2006.

[17] H. Topcuoglu, S. Hariri and M. Y. Wu, *Performance effective and low-complexity task scheduling for heterogeneous computing*, IEEE Trans. on Parallel and Distributed Systems, Volume 13, Issue 3, pp. 260-274, 2002.

[18] E. Ilavarasan and P. Thambidurai, *Low complexity performance effective task scheduling algorithm for heterogeneous computing environments*, Journal of Computer Sciences, Volume 3, Issue 2, pp. 94-103, 2007.

[19] J. Tordsson, R. Montero, R. Moreno-Vozmediano,and I. Llorente,*Cloud brokering mechanisms for optimized placement of virtual machines across multiple providers*, Future Generation Computer Systems, Volume 28, Issue 2, pp. 358-367, 2012.

[20] D. Breitgand, A. Marashini and J. Tordsson, *Policy-driven service placement optimization in federated clouds.*, Technical report, IBM Haifa Labs, 2011.

[21] R. V. den Bossche, K. Vanmechelen and J. Broeckhove, *Cost-optimal scheduling in hybrid iaas clouds for deadline constrained workloads*, In Proceedings of the 3rd IEEE international conference on cloud computing, pp. 228-235, 2010.

[22] R. Van den Bossche, K. Vanmechelen and J. Broeckhove, *Cost-Efficient Scheduling Heuristics for Deadline Constrained Workloads on Hybrid Clouds*, on Cloud Computing Technology and Science, IEEE International Conference, pp. 320-327, 2011.

[23] R. Buyya, C. S. Yeo, and S. Venugopal, *Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities*, Proc. Of the 10th IEEE International Conference on High Performance Computing and Communications, 2008, pp. 5-13.