A Multi-objective Optimization Approach to Workflow Scheduling in Clouds Considering Fault Recovery

Heyang Xu, Bo Yang, Weiwei Qi and Emmanuel Ahene
School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, Sichuan 611731, P. R. China
[e-mail: xuheyang124@126.com; yangbo@uestc.edu.cn; qiweiweiw@163.com; eahene@gmail.com]
*Corresponding author: Bo Yang

Received October 10, 2015; revised November 26, 2015; revised December 31, 2015; revised January 21, 2015; accepted January 25, 2016; published March 31, 2016

Abstract

Workflow scheduling is one of the challenging problems in cloud computing, especially when service reliability is considered. To improve cloud service reliability, fault tolerance techniques such as fault recovery can be employed. Practically, fault recovery has impact on the performance of workflow scheduling. Such impact deserves detailed research. Only few research works on workflow scheduling consider fault recovery and its impact. In this paper, we investigate the problem of workflow scheduling in clouds, considering the probability that cloud resources may fail during execution. We formulate this problem as a multi-objective optimization model. The first optimization objective is to minimize the overall completion time and the second one is to minimize the overall execution cost. Based on the proposed optimization model, we develop a heuristic-based algorithm called *Min-min based time and cost tradeoff (MTCT)*. We perform extensive simulations with four different real world scientific workflows to verify the validity of the proposed model and evaluate the performance of our algorithm. The results show that, as expected, fault recovery has significant impact on the two performance criteria, and the proposed MTCT algorithm is useful for real life workflow scheduling when both of the two optimization objectives are considered.

Keywords: Cloud computing, workflow scheduling, fault recovery, multi-objective optimization, heuristic-based algorithm.

1. Introduction

Cloud computing is a trend in distributed computing that delivers hardware infrastructure and software applications as services [1]. Its business model is based on the concept of paying only for what the users use, which overcome the limitations of the traditional software sales model [2]. Large-scale scientific workflows are usually represented as directed acyclic graphs in which computational tasks are represented by nodes and dependencies between tasks are represented by directed edges. They are an important class of applications [3]. Workflow scheduling in clouds basically involves the mapping of each task to an appropriate cloud resource and also ordering the tasks on each resoure so as to satisfy some performance criteria [4], such as the overall completion time and the overall execution cost. Workflow scheduling is however one of the challenging problems in the cloud computing. Since this problem is NP-complete, various heuristic-based algorithms have been proposed in literature [4-12].

1.1 Motivations

In recent years, workflow scheduling has been well studied by many researchers. However, one important aspect of the problem that has not been carefully addressed is that failures may happen on the resources. The probability of resource failures during the execution of task cannot be overlooked since it has direct impact on service performance. Our previous research works [13, 14] seem to be among the first ones that address this issue. To improve reliability, fault tolerance techniques such as fault recovery can be employed. Fault recovery adopts checkpoint and rollback/roll-forward scheme, which can enable a task to recover from an error and resume the executing of the task [14]. This technique can also be adopted to the problem of workflow scheduling in clouds and will certainly have significant impact on the two performance criteria, i.e., the overall completion time and the overall execution cost. Such impact deserves detailed research. However, existing researches on workflow scheduling in clouds rarely take the impact of fault recovery into account. Therefore, in this paper, we focus on addressing the issue of workflow scheduling in clouds considering fault recovery.

1.2 Related Work

Researchers investigated the problem of workflow scheduling in clouds from different points of views. Zeng et al. [6] proposed a budget-conscious workflow scheduling algorithm, ScaleStar. Arabnejad and Barbosa [7] proposed a heterogeneous budget constrained scheduling (HBCS) algorithm. The objective of this algorithm is to minimize the execution time of workflow application while guaranteeing the overall execution cost within a specified budget. Sakellariou et al. [8] proposed the Loss algorithm, which was basically targeted at minimizing the overall completion time under a user-defined budget constraint. Wu et al. [15] constructed analytical models to quantify the network performance of scientific workflows using cloud-based computing resources. They designed a critical-greedy algorithm to minimize the workflow end-to-end delay under a user-specified financial constraint. There is a rich body of research works on optimizing the completion time [17, 19-21]. However, in clouds, another important parameter, i.e., the overall execution cost, is rarely optimized in above research works.

In order to reduce the monetary cost of workflow execution, many cost-based algorithms have been proposed. For example, Abrishami et al. [4] proposed two workflow scheduling algorithms: a one-phase algorithm called *IaaS cloud partial critical paths (IC-PCP)* and a

two-phase algorithm called *IaaS cloud partial critical paths with deadline distribution (IC-PCPD2)*. Yuan et al [9] proposed a deadline distribution algorithm, *deadline early tree (DET)*, to optimize the cost of workflow execution under deadline constraint. Rodriguez and Buyya [11] proposed a resource provisioning and scientific workflows scheduling strategy. Their objective is to minimize the execution cost with ensuring that all the tasks are finished within their deadlines. Zhou and He [18] designed a resource provisioning system to simplify the optimization of monetary cost for scientific workflows. Some other research works [10, 22, 23] also aimed to find a solution that optimizes the monetary cost of workflow execution. However, all the above studies focus on the cost-based workflow scheduling algorithms and the completion time of a workflow is seen as a constraint, rather than an optimization objective, which is not enough for many modern workflow applications that need quick response [2, 15, 24].

Only a few existing research works [3, 25, 35] consider multi-objective optimization for workflow scheduling in clouds. More importantly, existing research works rarely consider the impact of fault recovery on certain performance criteria. Therefore, in this paper, we investigate the workflow scheduling problem by taking fault recovery into account and also considering two optimization objectives, i.e., minimizing both the overall completion time and the overall execution cost.

1.3 Contributions

We investigate the problem of workflow scheduling in clouds where the real world scenario of cloud resource failure during execution is considered. We adopt the fault recovery technique to address this problem. Our contributions in this paper can be summarized as follows.

First, we formulate the problem as a multi-objective optimization model. The first optimization objective is to minimize the overall completion time and the second one is to minimize the overall execution cost.

Second, we propose a heuristic-based algorithm called Min-min based time and cost tradeoff (MTCT) as a solution to the multi-objective optimization problem.

Finally, we perform two parts of experiments to verify the validity of our multi-objective optimization model and to evaluate the performance of our algorithm. The first experiment shows that, as expected, fault recovery has significant impact on the two performance criteria. In the second experiment, we compare our algorithm with other two popular algorithms in related research works.

The rest of this paper is organized as follows. Section 2 introduces the system models used in the paper. Section 3 presents the details of the proposed objective functions and the formulation of the multi-objective optimization model. Section 4 focuses on the details of the proposed MTCT algorithm. Experimental results and its analysis are given in Section 5. We conclude our paper in Section 6.

2. System Models

2.1 Workflow Model

Many complex applications in e-science and e-business can be modeled as workflows, which can be described by a directed acyclic graph (DAG) [9]. Denote a DAG as $G=(T,E)$, in which T is the set of $n(n\geq 1)$ tasks $\{t_1,t_2,...t_n\}$ and E is the set of dependencies. Each dependency $e_{i,j}=(t_i,t_j)$ represents a precedence constraint which indicates that t_i is a

parent task of t_j and t_j is a child task of t_i. The size of data required to be transmitted from task t_i to task t_j is denoted by $data[t_i, t_j]$. Based on this, a child task cannot be executed until all of its parent tasks are completed and the corresponding data are transmitted. In a given DAG, a task without any parent task is called an entry task and a task without any child task is called an exit task. Our algorithm makes use of a single entry task t_{entry} and a single exist task t_{exit}. The two tasks are however considered as dummy tasks and are set at the beginning and the end of the workflow respectively. The dummy tasks have zero exertion time and zero size of data and as such do not influence the process of scheduling the workflow. The length of task t_i in a given workflow is denoted by $length_i$, which can be achieved by some prediction techniques [26, 27]. **Table 1** provides a summary of the symbols used in this paper.

Table 1. Descriptions of symbols.

Symbols	Descriptions
G	the directed acyclic graph (DAG)
T	the set of tasks in a wokflow application
t_i	the *i*th task in a wokflow application
$pred(t_i)$	the set of parent tasks of task t_i
E	the set of dependencies in a wokflow application
$e_{i,j}$	the dependency between task t_i and task t_j
$data[t_i, t_j]$	the size of data required to be transmitted from task t_i to task t_j
$length_i$	the length of task t_i
R	the set of resources offered by the resource provider
r_j	the *j*th resource in R
ps_j	the processing speed of resource r_j
c_j	the cost of using resource r_j per unit time
ava_j	the available time of resource r_j
λ_j	the failure rate of resource r_j
μ_j	the recovery rate of resource r_j
bw	the average bandwidth between the resources
dt_{ij}	the data transfer time of dependency $e_{i,j}$
τ_{ij}	the estimated execution time of task t_i on resource r_j
AT_{ij}	the actual execution time of task t_i on resource r_j
$N_j(t)$	the total number of failures that occur on resource r_j during the time interval (0, t]
$RT_j^{(k)}$	the *k*th (*k*=1,2,…) recovery time on resource r_j
$RT_j(t)$	the total recovery time on resource r_j during (0, t]
$AFT(t_i, r_j)$	the actual finish time of task t_i on resource r_j

C_{ij}	the cost of executing task t_i on resource r_j
$M(G)$	the makespan function of a workflow G
$C(G)$	the overall execution cost function of a workflow G
β_i	the average finish time of task t_i on all resources
γ_i	the average execution cost of task t_i on all resources
$\varphi(t_i, r_j)$	the tradeoff metric function of executing task t_i on resource r_j

2.2 Resource Model

The cloud system can be seen as a resource provider, which offers multiple types of resources to its users [4]. Suppose that the resource provider offers $m\,(m \geq 1)$ resources, denoted by $R = \{r_1, r_2, ..., r_m\}$. These resources may have different CPU processing speeds, different memory sizes and different prices, which can be selected according to users' QoS requirements. ps_j and c_j represent the processing speed and the cost per time unit of resource r_j respectively. It is reasonable to assume that users need to pay higher prices for using the resources with higher processing speeds. ava_j represents the available time of resource r_j, which is defined as the earliest time at which resource r_j is ready for task execution. Suppose that all resources are available at the beginning, which means that the available time of all resources is zero initially. In this work, all the resources offered by the resource provider are assumed to be in the same physical region, so the average bandwidth between the resources, denoted by bw, is roughly the same [4, 11].

3. Problem Formulation

In this paper, we investigate the issue of workflow scheduling in clouds considering fault recovery. We take into account two performance criteria, namely the overall completion time and the overall execution cost. The overall completion time of a workflow is also called *makespan*. In the following, we give formal formulations of *the makespan function*, *the overall execution cost function* and *the multi-objective optimization model*.

Denote the data transfer time of a dependency $e_{i,j}$ as dt_{ij}. Since the average bandwidth between the resources is assumed to be equal, dt_{ij} only depends on the amount of data to be transferred between task t_i and task t_j, and it is independent of the resources that execute them. The only exception is when both tasks, t_i and t_j, are executed on the same resource, where dt_{ij} equals to zero. Let $r(t_i)$ denote the resource that executes task t_i, and dt_{ij} can be calculated as

$$dt_{ij} = \begin{cases} \dfrac{data[t_i, t_j]}{bw}, & r(t_i) \neq r(t_j) \\ 0, & otherwise \end{cases}. \tag{1}$$

Denote the estimated execution time of task t_i on resource r_j as τ_{ij}, which is given by

$$\tau_{ij} = \frac{length_i}{ps_j}. \tag{2}$$

Failures may happen on a resource when it is executing a task. If a failure is recoverable, then after some time (recovery time), the resource resumes the execution of the task [13]. In this paper, we only consider the case in which all failures are recoverable, since otherwise the task fails, which is out of the scope of this paper. Assume that the failure rate of a resource is a positive constant and the recovery times on a resource are independent and identically distributed random variables [14, 28]. Also, we assume that the occurrence of failures on resources, failure times, and recovery times are mutually s-independent [14]. Denote the failure rate of resource r_j as λ_j, which occurs in accordance with a Poisson process. Let $N_j(t)$ be the total number of failures that occur on resource r_j during the time interval $(0,t]$, then the probability of $N_j(t) = k$ ($k = 0,1,2,....$) can be given by

$$\Pr\{N_j(t) = k\} = \frac{(\lambda_j t)^k}{k!} e^{-\lambda_j t}, \quad k = 0,1,2,.... \tag{3}$$

The mean of $N_j(t)$ is

$$E[N_j(t)] = \lambda_j t. \tag{4}$$

Denote the recovery rate of resource r_j as μ_j. $RT_j^{(k)}$ is referred as the kth ($k = 1,2,....$) recovery time on resource r_j. The total recovery time on resource r_j during (0, t], denoted by $RT_j(t)$, can be given by

$$RT_j(t) = \sum_{k=1}^{N_j(t)} RT_j^{(k)}. \tag{5}$$

It can be seen that $RT_j(t)$ is a compound Poisson process, whose mean value is

$$E[RT_j(t)] = \frac{\lambda_j t}{\mu_j}. \tag{6}$$

After this, we can get the actual execution time (shown in Eq.(7)), which is different from the estimated execution time when fault recovery is adopted.

3.1 Makespan Function

Denote the actual execution time of task t_i on resource r_j as AT_{ij}, which is equal to the estimated execution time plus the total recovery time on resource r_j when task t_i is being executed. AT_{ij} can be given by

$$AT_{ij} = \tau_{ij} + RT_j(\tau_{ij}). \tag{7}$$

It can be seen that AT_{ij} is a random variable whose mean value is

$$E[AT_{ij}] = \frac{(\mu_j + \lambda_j)\tau_{ij}}{\mu_j}. \tag{8}$$

Let $\mathbf{X}=(x_{ij})_{n\times m}$ be the workflow allocation matrix. If task t_i is assigned to resource r_j, then $x_{ij}=1$; otherwise, $x_{ij}=0$. After task t_i is assigned to resource r_j, we can get its actual finish time $AFT(t_i,r_j)$, which can be given by

$$AFT(t_i,r_j)=\max\{ava_j,\max_{t_p\in pred(t_i)}\{AFT(t_p,r(t_p))+dt_{pi}\}\}+E[AT_{ij}], \quad (9)$$

where $pred(t_i)$ is the set of immediate predecessors of task t_i and ava_j is the available time of resource r_j, which can be obtained by

$$ava_j=\max_{\forall i\in\{i|r(t_i)=r_j\}}\{AFT(t_i,r_j)\}. \quad (10)$$

Let β_i be the average finish time of task t_i on all resources. β_i can be obtained by

$$\beta_i=\frac{\sum_{j=1}^{m}AFT(t_i,r_j)}{m}. \quad (11)$$

Since t_{entry} and t_{exit} are two dummy tasks, we have

$$AFT(t_{entry},r(t_{entry}))=0. \quad (12)$$

$$AFT(t_{exit},r(t_{exit}))=\max_{t_p\in pred(t_{exit})}\{AFT(t_p,r(t_p))\}. \quad (13)$$

Therefore, the makespan function of a workflow G (denoted by $M(G)$), which is the overall completion time of a workflow, is equal to the actual finish time of the exit task. So we have

$$M(G)=AFT(t_{exit},r(t_{exit})). \quad (14)$$

3.2 Overall Execution Cost Function

The overall execution cost of a workflow is the total expenses for executing all tasks of the workflow. Many cloud resource providers do not charge for the internal data transfer, so the data transfer cost is assumed to be zero in our cost function [4, 11]. Since the clouds try to realize the pay-as-you-go pricing model, the users only need to pay for what they use. Denote the cost of executing task t_i on resource r_j as C_{ij}, which can be given by

$$C_{ij}=c_j\cdot AT_{ij}. \quad (15)$$

Let γ_i be the average execution cost of task t_i on all resources. In essence, γ_i can be computed as

$$\gamma_i=\frac{\sum_{j=1}^{m}E[C_{ij}]}{m}. \quad (16)$$

Let $C(G)$ be the overall execution cost function of a workflow G, which is equal to the sum of the cost of executing all its tasks. $C(G)$ can be given by

$$C(G)=\sum_{i=1}^{n}\sum_{j=1}^{m}(x_{ij}\cdot C_{ij}). \quad (17)$$

It can be seen that $C(G)$ is a random variable whose mean value is

$$E[C(G)] = E[\sum_{i}^{n}\sum_{j}^{m}(x_{ij}\cdot C_{ij})]$$
$$= \sum_{i}^{n}\sum_{j}^{m}\left(x_{ij}\cdot c_{j}\cdot E[AT_{ij}]\right). \quad (18)$$

3.3 Multi-Objective Optimization Model

Usually, faster resources need more expenditure than slower ones for executing the same workflow. If a task is allocated on a faster resource, the completion time will be earlier and of course the cost will be higher; otherwise, the completion time will be later and of course the cost will be lower. Therefore, the two functions conflict with each other and the scheduler faces a time-cost tradeoff in selecting appropriate resources, which belongs to the multi-objective optimization (MOO) problem. Although we often adopt single-objective approach to address the MOO problem, there are some differences between MOO and single-objective optimization (SOO) [36, 37]. First, there are at least two distinct objectives in MOO, instead of only one objective in SOO; Second, MOO with conflicting objectives commonly results in a number of Pareto-optimal solutions, unlike the usual notion of only one optimal solution associated with SOO; Third, in MOO, the objective functions constitute a multidimensional space; and etc.

In this paper, we investigate the problem of workflow scheduling in clouds considering fault recovery: given a workflow application composed of a set of tasks with precedence constraints and a set of available resources, how to schedule each task to a suitable resource to optimize the performance criteria (the overall completion time and the overall execution cost) with the consideration of fault recovery. We formulate this problem as a MOO model

$$\min \quad M(G) \quad (19)$$
$$\min \quad E[C(G)] \quad (20)$$

Subject to:

$$\forall i \in \{1,\ldots,n\},\ \forall j \in \{1,\ldots,m\},\ x_{ij} \in \{0,1\}. \quad (21)$$

$$\forall i \in \{1,\ldots,n\},\ \sum_{j=1}^{m} x_{ij} = 1. \quad (22)$$

The optimization objectives are *minimizing the makespan*, (19), and *minimizing the mean value of the overall execution cost*, (20). The constraints of the proposed optimization model are (21) and (22), which indicate that a task can only be assigned to one resource.

4. Min-min based time and cost tradeoff (MTCT) algorithm

In the scheduling of each task, we concern two performance criteria, i.e., the actual finish time and the execution cost. As described in section 3.3, the finish time and execution cost are two conflicting criteria. Therefore, we propose a tradeoff between the finish time and the execution cost for scheduling a task. Let $\varphi(t_i, r_j)$ represent the tradeoff metric function of executing task t_i on resource r_j, and we have

$$\varphi(t_i, r_j) = \alpha \cdot \frac{AFT(t_i, r_j)}{\beta_i} + (1-\alpha)\frac{E[C_{ij}]}{\gamma_i}, \quad (23)$$

where $\alpha(0 \le \alpha \le 1)$ is the tradeoff factor. If a user prefers a shorter makespan, then α can be set to a value near 1; otherwise, α can be set to a value near 0. In our algorithm, α is set to 0.5, which means that we equally trade the makepan and the execution cost.

The developed algorithm, *Min-min based time and cost tradeoff (MTCT)*, is shown in Algorithm 1. Initially, two dummy tasks, t_{entry} and t_{exit}, and their corresponding dependencies are added to G and all tasks in the workflow are marked as *unscheduled* (lines 2-3, Algorithm 1). Then, the MTCT algorithm sets the actual finish time of task t_{entry} to zero and marks t_{entry} as *scheduled* (lines 4-5, Algorithm 1). Finally, the *ScheduleChildren* procedure is called for t_{entry} to schedule all its child tasks (line 6, Algorithm 1). The children tasks scheduling algorithm, as shown in Algorithm 2, will schedule all the workflow tasks.

Algorithm 1: The MTCT Scheduling Algorithm

Input: a workflow $G = (T, E)$ and set of resources $\{r_1, r_2, ...r_m\}$.

Output: the allocation matrix $\mathbf{X} = (x_{ij})_{n \times m}$.

1 **procedure** ScheduleWorkflow ($G = (T, E)$)

2 add t_{entry}, t_{exit} and their corresponding dependencies to G;

3 mark all tasks of G as *unscheduled*;

4 $AFT(t_{entry}, r(t_{entry})) \leftarrow 0$;

5 mark t_{entry} as *scheduled*;

6 call ScheduleChildren(t_{entry});

7 **end procedure**

The children tasks scheduling algorithm generally receives a *scheduled* task as input and schedules all its *unscheduled* children tasks. First, the *ScheduleChildren* procedure collects all the children tasks of the input task to the Children Tasks Set, CTS (lines 2-3, Algorithm 2). Afterwards, the while loop schedules tasks in CTS until CTS is empty (lines 4-19, Algorithm 2). In this process, the *ScheduleChildren* procedure always finds the longest task (t_i might as well) and judges whether all its parent tasks are *scheduled* or not (lines 5-18, Algorithm 2):

If all the parent tasks of t_i are scheduled, the *ScheduleChildren* procedure finds the resource (r_j might as well) with minimal value of tradeoff metric function as defined in Eq.(23) (lines 7-9, Algorithm 2). Afterwards, task t_i is assigned to resource r_j and marked as scheduled; as well, task t_i is removed from CTS and the available time of resource r_j changed (lines 10-14, Algorithm 2). After this, the *ScheduleChildren* procedure is recursively called to schedule the children tasks of t_i (line 15, Algorithm 2).

If one or more parent tasks of t_i are unscheduled, the *ScheduleChildren* procedure just removes task t_i from CTS (lines 16-17, Algorithm 2).

The above mentioned process is continued until all tasks are scheduled.

Algorithm 2: Children Tasks Scheduling (CTS) Algorithm

1 **procedure** ScheduleChildren(t)
2 $CTS \leftarrow \Phi$;
3 add all children tasks of t to the set CTS **;**
4 **while** ($CTS != \Phi$) **do**
5 $t_i \leftarrow$ get the longest task in CTS;
6 **if** (all parent tasks of t_i are *scheduled*)
7 **foreach** resource r_j in R **do**
8 find the resource with minimal value of tradeoff metric function

$$\varphi(t_i, r_j) = \alpha \cdot \frac{AFT(t_i, r_j)}{\beta_i} + (1-\alpha)\frac{E[C_{ij}]}{\gamma_i}$$

9 **end foreach**
10 assign task t_i to the scheduling queue of resource r_j;
11 set $x_{ij} = 1$;
12 mark task t_i as *scheduled*;
13 remove task t_i from CTS;
14 change the available time of resource r_j;
15 call ScheduleChildren(t_i);
16 **else**
17 remove task t_i from CTS;
18 **end if**
19 **end while**
20 **end procedure**

The proposed MTCT algorithm is based on the Min-Min algorithm. For each task, MTCT algorithm finds the resource with minimal value of tradeoff metric fuction as defined in (23). Since we equally trade the makepan and the execution cost in (23), MTCT algorithm can schedule each task to the resource which can finish the task as early as possible with relatively low execution cost. Afterwards, we analyze the time complexity of the proposed MTCT algorithm. Suppose that a workflow, DAG, has n tasks and e dependencies. For a given DAG, the number of dependencies could be at most n^2. The first part of the algorithm is to traverse the DAG to mark all its tasks as unscheduled. This process only needs a Depth-Fist or Breadth-First algorithm to traverse the nodes and the edges, so the time complexity of the first part is $O(n+e) \approx O(n^2)$. The second part is the *ScheduleChildren* procedure which is a recursive procedure. Actually, for each task in the DAG, *ScheduleChildren* procedure will be called once and only once. From Algorithm 2, we can easily find that time complexity of *ScheduleChildren* procedure is determined by the while loop (lines 4-19, Algorithm 2). For a

certain task in the DAG, its children tasks could be at most $(n-1)$. So in the worst case, the while loop will be invocated for $n \cdot (n-1)$ times. Meanwhile, the resource allocation operation (lines 6-17) needs to find the appropriate resource from m resources and this process will be invocated only once for each task. Therefore, the time complexity of *ScheduleChildren* procedure is $O(n^2 + n \cdot m)$. In practice, the maximum number of resources is usually less than n, so the time complexity of our algorithm is $O(n^2)$.

5. Experimental Studies

In this section, we present the experiments conducted in order to evaluate the performance of the proposed MTCT algorithm. The comparative algorithms, real world workflows used in our experiments, experimental setup and experimental results are shown in the following sections.

5.1. Comparative Algorithms

To evaluate the performance of the proposed MTCT algorithm, we compare it with other two algorithms, i.e., IaaS cloud partial critical paths (IC-PCP) algorithm [4] and Loss algorithm [8]. The Loss algorithm is proposed by Sakellariou et al. with the objective of minimizing the makespan of a workflow under a user-defined budget constraint. The Loss algorithm consists of two phases: In the first phase, it uses HEFT algorithm to generate an initial assignment. HEFT [17] is a well-known makespan minimization algorithm, which selects tasks in the descending order of their upward rank, and schedules them on the resource which can finish them as early as possible. The upward rank of a task is the length of a longest path from the task to the exit node [29]. In the second phase, the Loss algorithm judges whether the total cost of the initial assignment exceeds the budget or not. If the total cost of the initial assignment is less than the budget, then the initial assignment can be used straightaway to assign tasks to corresponding resources and the algorithm stops. Otherwise, it tries to refine the initial assignment by reassigning a task to a new resource, which has the minimum loss in execution time for the highest cost decrease. For this reason, it computes a Loss weight value for each task to each resource as follows:

$$LossWeight(t_i, r_j) = \frac{T_{new} - T_{old}}{C_{old} - C_{new}}, \tag{24}$$

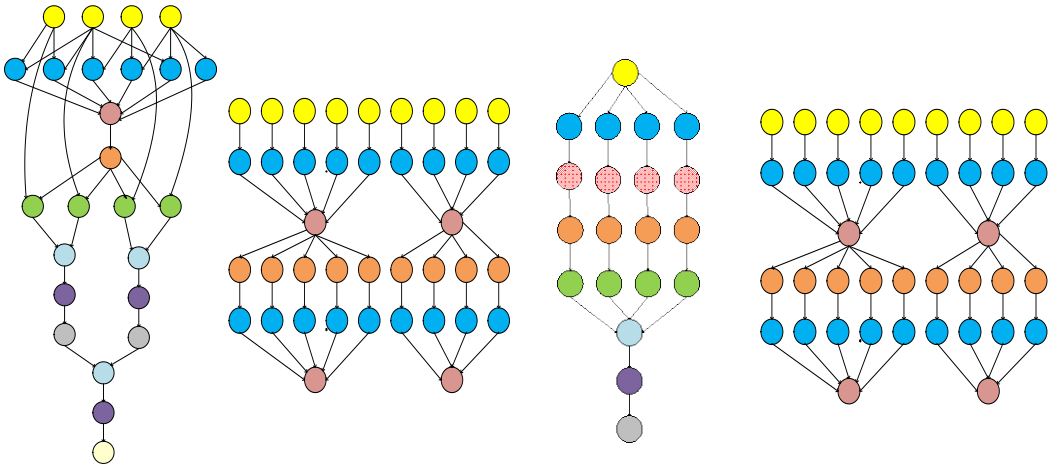
where T_{new} and C_{new} are the execution time and the cost of executing task t_i on resource r_j respectively, and T_{old} and C_{old} are the execution time and cost of executing task t_i on the initial assignment, respectively. The algorithm continues choosing the pair of task and resource which has the smallest value of LossWeight, and reassigning the task on the resource until the total cost is less than the budget.

Different from the Loss algorithm, the IC-PCP algorithm is proposed to conduct workflow scheduling in cloud environment with the target of minimizing the overall execution cost within a deadline constraint. First, the IC-PCP algorithm calculates the earliest start time (EST), the earliest finish time (EFT) and the latest finish time (LFT) for each task. Second, it finds a critical path associated to task t_{entry} and assigns the tasks on the path to the cheapest resource, which can meet the latest finish time requirements of the tasks on the critical path. Then, the ESTs, the EFTs and the LFTs of all unassigned tasks are updated. Finally, the IC-PCP algorithm recursively finds a partial critical path associated to the already assigned

task and assigns the tasks on the partial critical path to a resource until all tasks are assigned. According to [4], we set the deadline of the IC-PCP algorithm as two times of the makespan obtained by the fastest scheduling.

5.2. Experimental Workflows

Juve et al. [30] investigate the characterizations of six realistic workflows from diverse scientific applications, four of which are used in our experiments. They are Montage for astronomy, CyberShake for earthquake science, LIGO for gravitational physics and SIPHT for biology. **Fig. 1** shows the approximate structure of a small instance of each workflow used in our experiments. For each workflow, the tasks with the same color are of the same type. It can be seen that these four workflows have different structures, data and computational requirements. The full description of these workflows is presented in [30-32]. Four different sizes for each workflow in terms of total number of tasks are presented in [31], from where we download workflow cases in DAX format for our experiments. For each workflow, four different sizes, small, medium, large and extra-large, are used in our experiments. Small workflows have about 30 tasks in one workflow application whereas medium ones have about 50, large ones 100 and extra-large ones about 1000. The detail description of the workflows used in our experiments is shown in **Table 2**. The runtimes of all tasks in the original workflows are relatively short, so we enlarge the size of each workflow task 1000 times to make the workflows large enough.



(a) Montage (b) CyberShake (c) Epigenomics (d) LIGO

Fig. 1. The structures of four different realistic scientific workflows.

Table 2. Details of workflows used in experiments.

workflows	the number of tasks in different workflow sizes			
	Small	Medium	Large	XLarge
Montage	25	50	100	1000
CyberShake	30	50	100	1000
Epigenomics	24	46	100	1000
LIGO	30	50	100	977

5. 3. Experimental Setup

All the experiments are performed on a Pentium(R) Dual-Core processor with a speed of 2.8GHz and memory of 4GB. The CloudSim toolkit [33] is used to simulate a cloud system, which consists of 20 heterogeneous physical servers with different processor speeds, prices, failure rates and recovery rates. Each physical server has only one CPU core and executes tasks sequentially. For the processor speeds and prices, we apply the similar method to generate the values as used in [4]. The processor speed of each server is uniformly distributed within the range [100, 1000] with the average speed of 550 MIPS. The price of a server has roughly linear relationship with its processor speed, which makes sure that a faster server needs more execution cost than a slower server for executing a same task. The failure rates of the physical servers are randomly generated from the interval of [0.01, 0.1] [14, 34]. The recovery rates of these servers are randomly generated from the interval of [0.05, 0.15]. The average bandwidth between the servers is set to 20 Mbps as used in [4, 35]. Each experiment is run 1000 times and the results presented in the figures are the mean value of the results obtained by all the 1000 experiments.

5.4. Experimental results

In this section, we present the results obtained by different algorithms. In the first experiment, we evaluate the impact of resource failures on the two performance criteria. In the second experiment, we use the measured metrics, *the normalized cost* (*NC*), *the normalized makespan* (*NM*), to evaluate the performance of the proposed algorithm

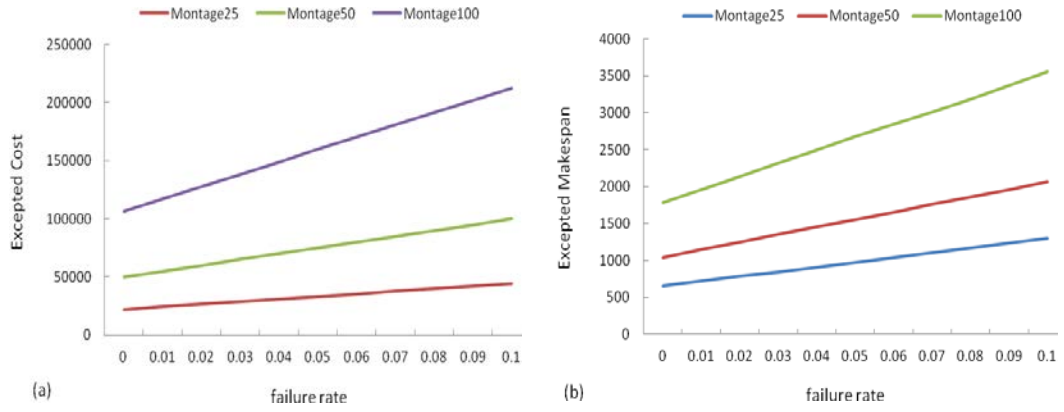


Fig. 2. The results obtained by different failure rates of scheduling Montage workflows with MTCT algorithm when μ=0.1.

5.4.1. Experiment 1

This experiment evaluates the impact of resource failures on the two performance criteria. **Fig. 2** shows the results obtained by MTCT algorithm with different failure rates of scheduling Montage workflows when all the resources' recovery rate is 0.1. It can be seen that when the recovery rate is set to a constant, the expected cost and expected makespan smoothly increase with the increasing of resources' failure rate. When the failure rate increases from 0.01 to 0.1, the two measured metrics increase almost two times. Likewise, **Fig. 3** shows the results obtained by MTCT algorithm with different recovery rates of scheduling Montage workflows when all the resources' failure rate is 0.05. As it is seen from **Fig. 3**, with fixed failure rate, the expected total cost and expected makespan decrease significantly when the recovery rate

increases from 0.05 to 0.5. Therefore, it can be concluded that fault recovery, as expected, has significant impact on the performance criteria for workflow scheduling in clouds.

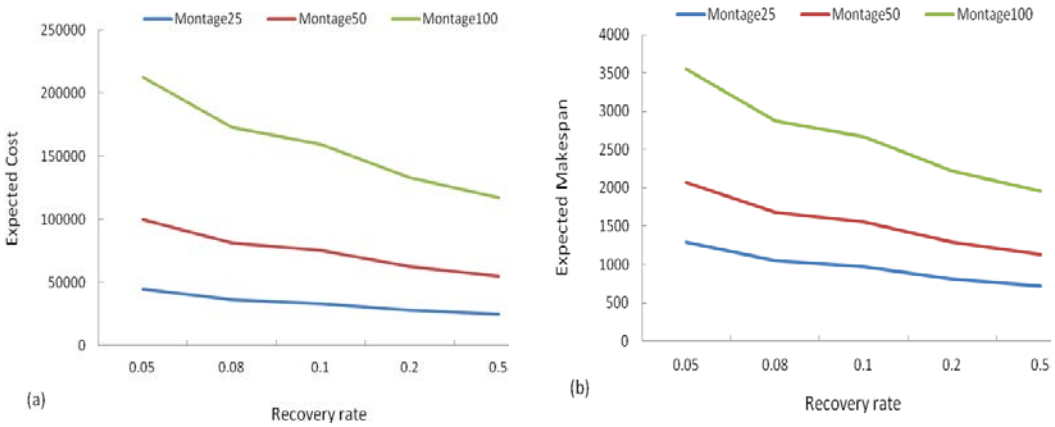


Fig. 3. The results obtained by different recovery rates of scheduling Montage workflows with MTCT algorithm when λ=0.05.

5.4.1. Experiment 1

This experiment evaluates the impact of resource failures on the two performance criteria. **Fig. 2** shows the results obtained by MTCT algorithm with different failure rates of scheduling Montage workflows when all the resources' recovery rate is 0.1. It can be seen that when the recovery rate is set to a constant, the expected cost and expected makespan smoothly increase with the increasing of resources' failure rate. When the failure rate increases from 0.01 to 0.1, the two measured metrics increase almost two times. Likewise, **Fig. 3** shows the results obtained by MTCT algorithm with different recovery rates of scheduling Montage workflows when all the resources' failure rate is 0.05. As it is seen from **Fig. 3**, with fixed failure rate, the expected total cost and expected makespan decrease significantly when the recovery rate increases from 0.05 to 0.5. Therefore, it can be concluded that fault recovery, as expected, has significant impact on the performance criteria for workflow scheduling in clouds.

5.4.2. Experiment 2

In this section, we evaluate the performance of the proposed algorithm by using the following two measured metrics, *the normalized cost* (*NC*) [9, 35] and *the normalized makespan* (*NM*).

The normalized cost of a workflow execution, denoted by NC (as shown in Eq.(25)), is defined as the total execution cost obtained by the corresponding algorithm divided by C_c, which is the cost of executing the same workflow on the cheapest computation resource. The smaller the value of NC, the less expenditure a user needs to pay for executing its workflow, and also the better the result.

$$NC = \frac{total \quad execution \quad cost}{C_c}. \tag{25}$$

Similarly, the normalized makespan of a workflow execution, denoted by NM (as shown in Eq.(25)), is defined as the makespan obtained by the corresponding algorithm divided by M_{Min}, which is the makespan obtained by the fastest scheduling algorithm, i.e. the Min-min

algorithm, for executing the same workflow. The smaller the value of NM, the shorter completion time of the workflow is, and also the better the result.

$$NM = \frac{makespan}{M_{Min}}. \tag{26}$$

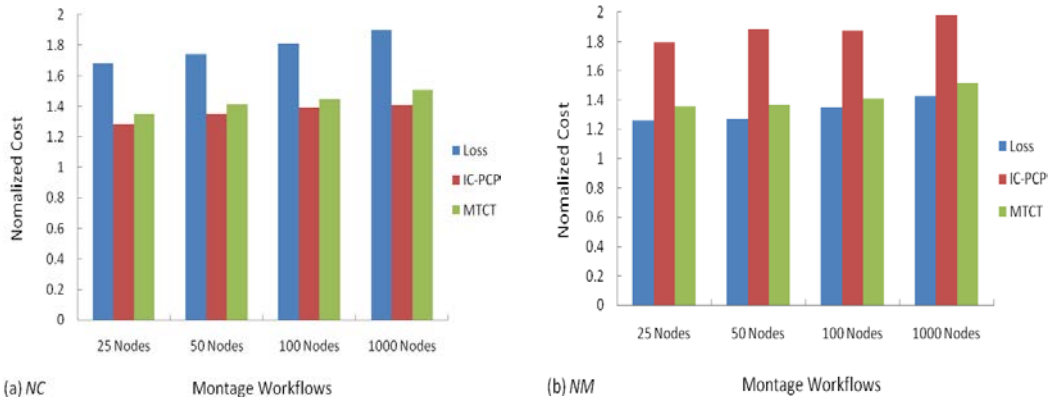


Fig. 4. Comparison on NC and NM with Montage workflows.

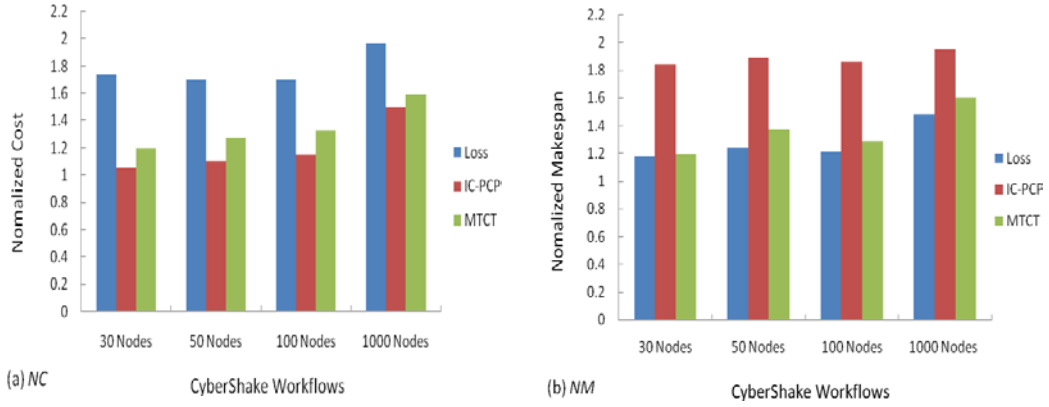


Fig. 5. Comparison on NC and NM with CyberShake workflows.

Fig. 4 - Fig. 7 show the results of the NC and NM obtained by the three algorithms for different kinds of workflows with different sizes. From **Fig. 4(a)**, it can be seen that IC-PCP algorithm can obtain the best NC for Montage workflows, because this algorithm always assigns tasks on a partial critical path to the resource, which can meet the latest finish time of the tasks with the lowest cost. However, the NM obtained by IC-PCP algorithm for Montage workflows is the worst among the three comparative algorithms. From **Fig. 4(b)**, it can be seen that Loss algorithm can obtain the best NM for Montage workflows. However, the NC obtained by Loss algorithm for Montage workflows is the worst. Compared to the Loss algorithm, the proposed MTCT algorithm can decrease NC by 29.2% with only a 6.9% increase in NM, using Montage workflows (**Fig. 4(a)**). Also, compared to IC-PCP algorithm, the proposed MTCT algorithm can decrease NM by 28.9% with only a 5.1% increase in NC (**Fig. 4(b)**). Moreover, as shown in **Fig. 5 - Fig. 7**, using the CyberShake, Epigenomics and LIGO workflows to test the comparative algorithms, the results turn out to be similar to the

case where Montage workflows is used. From the above comparisons, it can be obvious to deduce that, among the three algorithms, Loss algorithm can obtain the shortest makespan with relatively high execution cost; IC-PCP algorithm can obtain the lowest execution cost with relatively long makespan and the proposed MTCT algorithm can obtain both relatively low execution cost and relatively short makespan. In essence, the proposed MTCT algorithm is useful for real life workflow scheduling with the objective of optimizing both the overall execution cost and the makespan.

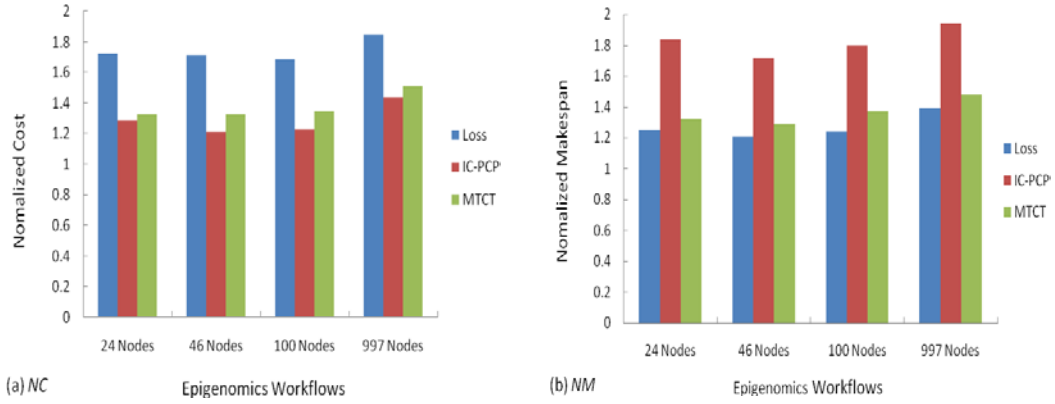


Fig. 6. Comparison on NC and NM with Epigenomics workflows.

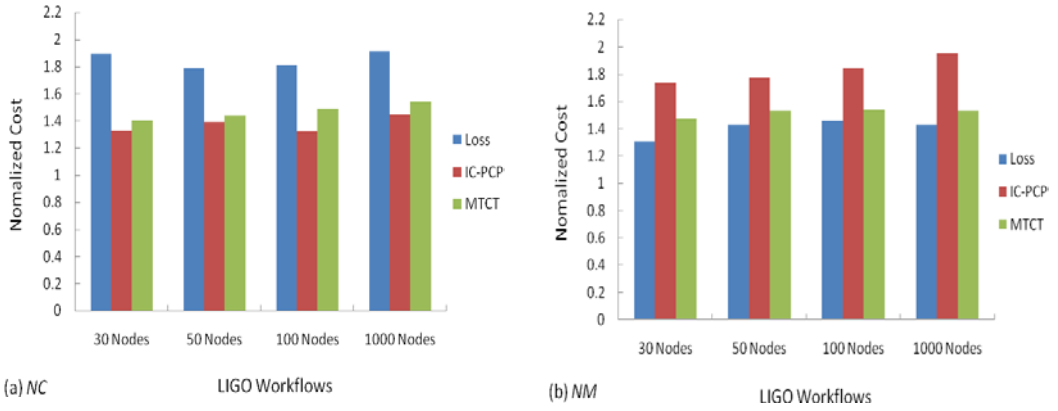


Fig. 7. Comparison on NC and NM with LIGO workflows.

6. Conclusion and Future Work

This paper investigates the problem of workflow scheduling in clouds considering fault recovery. The scenario is modeled as a multi-objective optimization problem which aims to minimize the overall completion time and the overall execution cost. Four well-known real world workflow applications are chosen to evaluate the performance of the proposed algorithm and the results show that the proposed MTCT algorithm is a better choice when both the overall execution cost and the makespan are considered.

As future work, we will consider the situation that not all failures are recoverable and tasks may fail. Moreover, we may also consider the probability of failures on communication

channels.

Acknowledgements

This work is supported by the Fundamental Research Funds for the Central Universities (Grant No.: ZYGX2013J066) and Sichuan Provincial Project of International Scientific and Technical Exchange and Research Collaboration Programs.

References

[1] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, M. Zaharia, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50-58, 2010. Article (CrossRef Link).

[2] X. Yang, B. Nasser, M. Surridge, S. Middleton, "A business-oriented Cloud federation model for real-time applications," *Future Generation Computer Systems*, vol. 28, no. 8, pp. 1158-1167, 2012. Article (CrossRef Link).

[3] K. Bessai, S. Youcef, A. Oulamara, C. Godart, "Bi-criteria strategies for business processes scheduling in cloud environments with fairness metrics," in *Proc. of IEEE Seventh International Conference on Research Challenges in Information Science*, pp. 1-10, 2013. Article (CrossRef Link).

[4] S. Abrishami, M. Naghibzadeh, D.H. Epema, "Deadline-constrained workflow scheduling algorithms for infrastructure as a service clouds," *Future Generation Computer Systems*, vol. 29, no. 1, pp. 158-169, 2013. Article (CrossRef Link).

[5] M. Malawski, G. Juve, E. Deelman, J. Nabrzyski, "Algorithms for cost- and deadline-constrained provisioning for scientific workflow ensembles in IaaS clouds," *Future Generation Computer Systems*, vol.48, pp. 1-18, 2015. Article (CrossRef Link).

[6] L. Zeng, B. Veeravalli, X. Li, "ScaleStar: Budget conscious scheduling precedence-constrained many-task workflow applications in cloud," in *Proc. of IEEE 26th International Conference on Advanced In-formation Networking and Applications*, pp. 534-541, 2012. Article (CrossRef Link).

[7] H. Arabnejad, J.G. Barbosa, "A budget constrained scheduling algorithm for workflow applications," *Journal of Grid Computing*, vol. 12, no.4, pp. 665-679, 2014. Article (CrossRef Link).

[8] R. Sakellariou, H. Zhao, E. Tsiakkouri, M.D. Dikaiakos, "Scheduling workflows with budget constraints," *Integrated Research in Grid Computing*, pp. 189-202, 2007. Article (CrossRef Link).

[9] Y. Yuan, X. Li, Q. Wang, X. Zhu, "Deadline division-based heuristic for cost optimization in workflow scheduling," *Information Sciences*, vol. 179, no. 15, pp. 2562-2575, 2009. Article (CrossRef Link).

[10] M. Mao, M. Humphrey, "Auto-scaling to minimize cost and meet application deadlines in cloud workflows," in *Proc. of 2011 Inter-national Conference for High Performance Computing, Networking, Storage and Analysis*. Article (CrossRef Link).

[11] M.A. Rodriguez, R. Buyya, "Deadline based resource provisioning and scheduling algorithm for scientific workflows on clouds," *IEEE Transactions on Cloud Computing*, vol. 2, no. 2, pp. 222-235, 2014. Article (CrossRef Link).

[12] R.N. Calheiros, R. Buyya, "Meeting deadlines of scientific workflows in public clouds with tasks replication," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 7, pp. 1787-1796, 2013. Article (CrossRef Link).

[13] B. Yang, F. Tan, Y. Dai, S. Guo, "Performance evaluation of cloud service considering fault recovery," in *Proc. of the First International Conference on Cloud Computing*, pp. 571-576, 2009. Article (CrossRef Link).

[14] B. Yang, F. Tan, Y. Dai, "Performance evaluation of cloud service considering fault recovery," *The Journal of Supercomputing*, vol. 65, no. 1, pp. 426-444, 2013. Article (CrossRef Link).

[15] C.Q. Wu, X. Lin, D. Yu, W. Xu, L. Li, "End-to-end delay minimization for scientific workflows in clouds under budget constraint," *IEEE Transactions on Cloud Computing*, vol. 3, no. 2, pp. 169-181, 2015. Article (CrossRef Link).
[16] H. Xu, B. Yang, "An incentive-based heuristic job scheduling algorithm for utility grids," *Future Generation Computer Systems*, vol. 49, pp. 1-7, 2015. Article (CrossRef Link).
[17] H. Topcuoglu, S. Hariri, M. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, no. 3, pp. 260-274, 2002. Article (CrossRef Link).
[18] A.C. Zhou, B. He, "Simplified resource provisioning for workflows in IaaS clouds," in *Proc. of IEEE 6th International Conference on Cloud Computing Technology and Science*, pp. 650-655, 2014. Article (CrossRef Link).
[19] J. Zhang, J. Luo, F. Dong, "Scheduling of scientific workflow in non-dedicated heterogeneous multicluster platform," *Journal of Systems and Software*, vol. 86, no. 7, pp. 1806-1818, 2013. Article (CrossRef Link).
[20] C. Lin, S. Lu, "Scheduling scientific workflows elastically for cloud computing," in *Proc. of IEEE 4th International Conference on Cloud Computing*, pp. 246-247, 2011. Article (CrossRef Link).
[21] D. Jung, T. Suh, H. Yu, J. Gil, "A workflow scheduling technique using genetic algorithm in spot instance-based cloud," *KSII Transactions on Internet and Information Systems*, vol.8, no. 9, pp. 3126-3145, 2014. Article (CrossRef Link).
[22] J. Sahni, D. Vidyarthi, "A cost-effective deadline-constrained dynamic scheduling algorithm for scientific workflows in a cloud environment," *IEEE Transactions on Cloud Computing*, preprint, 2015. Article (CrossRef Link).
[23] D. Poola, K. Ramamohanarao, R. Buyya, "Fault-tolerant workflow scheduling using spot instances on clouds," *Procedia Computer Science*, vol. 29, pp. 523-533, 2014. Article (CrossRef Link).
[24] Y.W. Ahn, A.M.K. Cheng, J. Baek, M. Jo, H.H. Chen, "An auto-scaling mechanism for virtual resources to support mobile, pervasive, real-time healthcare applications in cloud computing," *IEEE Network*, vol. 27, pp. 62-68, 2013. Article (CrossRef Link).
[25] D. Poola, S.K. Garg, R.Buyya, Y. Yang, K. Ramamohanarao, "Robust scheduling of scientific workflows with deadline and budget constraints in clouds," in *Proc. of IEEE 28th International Conference on Advanced Information Networking and Applications*, pp. 858-865, 2014. Article (CrossRef Link).
[26] S. Jang, V. Taylor, X. Wu, M. Prajugo, E. Deelman, G. Mehta, K. Vahi, "Performance prediction-based versus load-based site selection: Quantifying the difference," in *Proc. of the 18th International Conference on Parallel and Distributed Computing Systems*, pp. 148-153, 2005. Article (CrossRef Link)
[27] M. De Felice, X. Yao, "Short-term load forecasting with neural network ensembles: A comparative study," *IEEE Computational Intelligence Magazine*, vol. 6, no. 3, pp. 47-56, 2011. Article (CrossRef Link).
[28] Y.S Dai, G. Levitin, K.S. Trivedi, "Performance and reliability of tree-structured grid services considering data dependence and failure correlation," *IEEE Transactions on Computers*, vol. 56, no. 7, pp. 925–936, 2007. Article (CrossRef Link).
[29] Y.K. Kwok, I. Ahmad, "Static scheduling algorithms for allocating directed task graphs to multiprocessors," *ACM Computing Surveys*, vol. 34, no. 4, pp. 406-470, 1999. Article (CrossRef Link).
[30] G. Juve, A. Chervenak, E. Deelman, S. Bharathi, "Characterizing and profiling scientific workflows," *Future Generation Computer Systems*, vol. 29, no. 3, pp. 682-692, 2013. Article (CrossRef Link).
[31] Workflow Generator, https://confluence.pegasus.isi.edu/display/pegasus/WorkflowGenerator, 2014.
[32] S. Bharathi, A. Chervenak, E. Deelman, G. Mehta, M. Su, K. Vahi, "Characterization of scientific workflows," in *Proc. of Third Workshop on Workflows in Support of Large Scale Science*, pp. 1-10, 2008. Article (CrossRef Link).

[33] R.N. Calheiros, R. Ranjan, A. Beloglazov, C. De Rose, 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, pp. 23-50, 2011. Article (CrossRef Link).
[34] B. Yang, H. Hu, S. Guo, "Cost-oriented task allocation and hardware redundancy policies in heterogeneous distributed computing systems considering software reliability," *Computers & Industrial Engineering*, vol. 56, no. 4, pp. 1687-1696, 2009. Article (CrossRef Link).
[35] A. Verma, S. Kaushal, "Bi-criteria priority based particle swarm optimization workflow scheduling algorithm for cloud," in *Proc. of 2014 Recent Advances in Engineering and Computational Sciences*, pp. 1-6, 2014. Article (CrossRef Link).
[36] R.T. Marler, J.S. Arora, "Survey of multi-objective optimization methods for engineering," *Structural and Multidisciplinary Optimization*, vol. 26, pp. 369-395, 2004. Article (CrossRef Link).
[37] K. Deb. Multi-objective optimization. in Search Methodologies, pp. 403-449, Springer, 2014. Article (CrossRef Link).

Heyang Xu is a Ph.D. candidate in the School of Computer Science and Engineering (SCSE), University of Electronic Science and Technology of China (UESTC), Chengdu, China. He received the B.E. degree in Computer Science from Yunnan Police Officer Academy, Kunming, China, in 2010. His research interests include grid computing and cloud computing. He has published a regular paper in Future Generation Computer Systems and a paper in the 2nd IEEE International Conference on Parallel, Distributed and Grid Computing.

Bo Yang received the B.E. (1995), and the M.E. (1998) degrees from Xi'an Jiaotong University, Xi'an, China; and the Ph.D. (2002) from National University of Singapore, Singapore. He is a Professor and the Deputy Head of Collaborative Autonomic Computing Laboratory, SCSE, UESTC. His research interests include distributed/grid/cloud computing, data mining, and software and system reliability engineering. He has published over 50 research papers in refereed academic journals and conferences. He is on the Editorial Board of six international journals. He served as Program Chair of The 8th IEEE International Conference on Dependable, Autonomic and Secure Computing (IEEE DASC 2009); Program Chair of The 2011 International Conference on Cloud and Service Computing (CSC 2011); Program Vice-Chair of The 12th IEEE International Conference on High Performance and Communications (IEEE HPCC 2010); General Chair of The 2010 International Workshop on Knowledge and Data Engineering in Web-based Learning (IWKDEWL'10); and Program Committee Member of over 30 international conferences or workshops. He is a senior member of IEEE, China Computer Federation (CCF), Chinese Institute of Electronics (CIE), and member of CCF Technical Committee on Collaborative Computing (CCF TCCC).

Weiwei Qi is an M.E. candidate in SCSE, UESTC, Chengdu, China. He received the B.E. degree in Computer Science from UESTC, Chengdu, China, 2014. His research interests include cloud computing and data mining.

Emmanuel Ahene is an M.E. candidate in SCSE, UESTC, Chengdu, China. He received the B.S. degree in Computer Science from the University for Development Studies, Ghana, in 2012. His research interests include cloud computing and information security.