



Sharif University of Technology

Scientia Iranica

Transactions D: Computer Science &amp; Engineering and Electrical Engineering

www.sciencedirect.com



Research note

## Deadline-constrained workflow scheduling in software as a service Cloud

S. Abrishami\*, M. Naghibzadeh

Department of Computer Engineering, Ferdowsi University of Mashhad, Mashhad, P.O. Box 91755-1111, Iran

Received 27 February 2011; revised 19 June 2011; accepted 14 November 2011

### KEYWORDS

Cloud computing;  
SaaS Clouds;  
Grid computing;  
Workflow scheduling;  
QoS-based scheduling.

**Abstract** The advent of Cloud computing as a new model of service provisioning in distributed systems, encourages researchers to investigate its benefits and drawbacks in executing scientific applications such as workflows. In this model, the users request for available services according to their desired Quality of Service, and they are charged on a pay-per-use basis. One of the most challenging problems in Clouds is workflow scheduling, i.e., the problem of satisfying the QoS of the user as well as minimizing the cost of workflow execution. In this paper, we propose a new QoS-based workflow scheduling algorithm based on a novel concept called Partial Critical Paths (PCP), which tries to minimize the cost of workflow execution while meeting a user-defined deadline. This algorithm recursively schedules the partial critical paths ending at previously scheduled tasks. The simulation results show that the performance of our algorithm is very promising.

© 2012 Sharif University of Technology. Production and hosting by Elsevier B.V. All rights reserved.

### 1. Introduction

Cloud computing [1] is the latest emerging trend in distributed computing that delivers hardware infrastructures and software applications as services. The users can consume these services based on a *Service Level Agreement (SLA)*, which defines their required Quality of Service (QoS) parameters on a pay-per-use basis. Although there are many papers that address the problem of scheduling in traditional distributed systems, like Grids, there are only a few works on this problem in Clouds. The multi-objective nature of the scheduling problem in Clouds makes it difficult to solve, especially in the case of complex jobs like workflows. This has led most researchers to use time-consuming meta-heuristic approaches, instead of fast heuristic methods. In this paper, we propose a new heuristic algorithm for scheduling workflows in Clouds, and we evaluate its performance on some well-known scientific workflows.

Recently, many researchers have considered the benefits of using Cloud computing for scientific applications [2–4]. Using virtualization technologies, Clouds can offer the users a wide range of services from hardware to the application level. Currently, these services are categorized into three major classes: Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). IaaS Clouds, like Amazon, provide virtualized hardware and storage on top of which the users can deploy their own applications and services. PaaS Clouds, like Microsoft Azure, provide an application development environment in which the users can implement and run applications on the Cloud. According to [5], there are two types of Cloud, which deliver software applications to the users. The first group offers an entire application as a service to the end users, which can be used without any changes or customization. Examples of these Clouds are Google office automation services like Google Document or Google Calendar. The second group provides rudimentary web services to the users (known as on-demand web services), which can be used to build more complex applications. Xignite and Strikelron offer web services hosted on a Cloud on a pay per use basis. We have used the second group in our SaaS Cloud model.

Workflows constitute a common model for describing a wide range of scientific applications in distributed systems. Usually, a workflow is described by a Directed Acyclic Graph (DAG) in which each computational task is represented by a

\* Corresponding author.

E-mail addresses: s-abrishami@um.ac.ir (S. Abrishami), naghibzadeh@um.ac.ir (M. Naghibzadeh).

Peer review under responsibility of Sharif University of Technology.



Production and hosting by Elsevier

node, and each data or control dependency between tasks is represented by a directed edge between the corresponding nodes. Due to the importance of workflow applications, many Grid projects, such as Pegasus [6], ASKALON [7] and GrADS [8], have designed workflow management systems to define, manage and execute workflows on the Grid. Having been successful on Grids, researchers are now investigating the running of large scientific workflows on Clouds [4]. Furthermore, the latest version of Grid workflow management systems, like Pegasus, VGrADS [9] and ASKALON [10], also support running scientific workflows on Clouds.

Workflow scheduling is the problem of mapping each task to a suitable resource and of ordering the tasks on each resource to satisfy some performance criterion. As task scheduling is a well-known NP-complete problem [11], many heuristic methods have been proposed for homogeneous [12] and heterogeneous distributed systems like Grids [13,14]. These scheduling methods try to minimize the execution time (makespan) of the workflows and, as such, are suitable for community Grids. Most current workflow management systems, like the ones above mentioned, use such scheduling methods. However, in Clouds, there are many other potential QoS attributes besides execution time, like reliability, security, availability and so on. Besides, stricter QoS attributes mean higher prices for services. Therefore, the scheduler faces a QoS-cost tradeoff in selecting appropriate services, which belongs to the *multi-objective optimization problems* family. There are several existing approaches to the problem of multi-objective scheduling [15]. One method is to find *pareto optimal* solutions, and let the user select the best schedule according to his requirements. The problem is that the pareto sets are usually very large and hard to examine. Another common method is to assign a weight to each scheduling criterion, and optimize the weighted sum of these criteria. However, in most cases, the weight assignment is not a straightforward process for users. Due to the complexities of the development of a general multi-objective scheduling algorithm, many researchers try to propose bi-criteria scheduling algorithms. In most bi-criteria scheduling algorithms, the user specifies a limitation for one criterion (deadline or budget constraints), and the algorithm tries to optimize the other criterion under this constraint. This is a convenient way for the users to express their requirements, and a helpful method for the researchers to simplify the problem and propose fast and high performance solutions.

In our previous paper, we proposed a QoS-based workflow scheduling algorithm on utility Grids, called the *Partial Critical Paths* (PCP) [16]. In this paper, we propose the SaaS Cloud Partial Critical Paths (SC-PCP) algorithm, which is an extension of the previous one for the SaaS Clouds. The objective function of the SC-PCP algorithm is to create a schedule that minimizes the total execution cost of a workflow, while satisfying a user-defined deadline for the total execution time. First, the SC-PCP algorithm tries to schedule the (overall) critical path of the workflow, such that it is completed before the user deadline, and execution cost is minimized. Then, it finds the *partial critical path* to each scheduled task on the critical path and executes the same procedure in a recursive manner.

The remainder of the paper is organized as follows. Section 2 describes our system model including the application model the Cloud model and the objective function. The SC-PCP scheduling algorithm is explained in Section 3. A performance evaluation is presented in Section 4. Section 5 reviews related work and Section 6 concludes.

## 2. Scheduling system model

The proposed scheduling system model consists of an application model, a Cloud model, and a performance criterion for scheduling. An application is modeled by a directed acyclic graph  $w(T, E)$ , where  $T$  is a set of  $n$  tasks  $\{t_1, t_2, \dots, t_n\}$ , and  $E$  is a set of dependencies. Each dependency,  $e_{i,j} = (t_i, t_j)$ , represents a precedence constraint, which indicates that task  $t_i$  should complete execution before task  $t_j$  can start. In a given task graph, a task without any parent is called an *entry task*, and a task without any child is called an *exit task*. As our algorithm requires a single entry and a single exit task, we always add two dummy tasks;  $t_{entry}$  and  $t_{exit}$ , to the beginning and end of the workflow, respectively. These dummy tasks have zero execution time and are connected with zero-weight dependencies to the actual entry and exit tasks.

Our Cloud model consists of an SaaS provider which provides web services to its clients and is capable of executing scientific workflows. The service provider offers several services with different QoS for each task of every workflow. We assume that each workflow task,  $t_i$ , can be processed by  $m_i$  services,  $S_i = \{s_{i,1}, s_{i,2}, \dots, s_{i,m_i}\}$ , with different QoS attributes. There are many QoS attributes for services, like execution time, cost, reliability, security and so on. In this study, we use the most important ones, execution time and cost, for our scheduling model. Furthermore, the pricing model is *per transaction pricing*, which charges users based on the number of completed transactions. In this context, a completed transaction means a successfully finished workflow task. Some real Clouds, like Xignite and Strikelron, use this pricing model. The price of a service usually depends on its execution time, i.e. shorter execution times are more expensive. We assume that the services of each task are sorted in an increasing order according to their execution times, i.e.  $s_{i,j}$  is faster (and more expensive) than  $s_{i,j+1}$ . Besides,  $ET(t_i, s)$  and  $EC(t_i, s)$  are defined as the execution time and the execution cost of processing task  $t_i$  on service  $s$ , respectively. Finally, we assume that there is no limitation on using services and we can schedule a task on every potential service, at any time. This assumption is based on an important feature of current commercial public Clouds like Amazon, the illusion of unlimited resources [2]. It means that the user can ask for every service of the Cloud whenever he needs it, and he will certainly (or with a very high possibility) obtain that service. Of course, this assumption may fail in the future, when Clouds become more popular.

There is another source of time and money consumption: transferring data between tasks. All tasks are assumed to use a shared storage service (such as Amazon Simple Storage Service [17]) to send and receive the intermediate data. Besides, all computation and storage services of a service provider are assumed to be in the same physical region (such as Amazon Regions), so the average bandwidth between the computation services and the storage service is roughly equal. With this assumption, the data transfer time of a dependency  $e_{i,j}$  only depends on the amount of data to be transferred between corresponding tasks, and it is independent of the services which execute them. Therefore,  $TT(e_{ij})$  is defined as the data transfer time of a dependency,  $e_{i,j}$ , independent of the selected service for  $t_i$  and  $t_j$ . Furthermore, the internal data transfer is free in most real Clouds, so the data transfer cost is assumed to be zero in our model. Of course, the service provider charges the clients for using the storage service based on the amount of allocated volume, and possibly for the number of I/O transactions from/to outside the Cloud. Since these parameters are constant for

متن کامل مقاله

دریافت فوری ←

**ISI**Articles

مرجع مقالات تخصصی ایران

- ✓ امکان دانلود نسخه تمام متن مقالات انگلیسی
- ✓ امکان دانلود نسخه ترجمه شده مقالات
- ✓ پذیرش سفارش ترجمه تخصصی
- ✓ امکان جستجو در آرشیو جامعی از صدها موضوع و هزاران مقاله
- ✓ امکان دانلود رایگان ۲ صفحه اول هر مقاله
- ✓ امکان پرداخت اینترنتی با کلیه کارت های عضو شتاب
- ✓ دانلود فوری مقاله پس از پرداخت آنلاین
- ✓ پشتیبانی کامل خرید با بهره مندی از سیستم هوشمند رهگیری سفارشات