Future Generation Computer Systems 78 (2018) 402–412

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

# Reprint of "Multi-QoS constrained and Profit-aware scheduling approach for concurrent workflows on heterogeneous systems"

Hamid Arabnejad, Jorge G. Barbosa *

*LIACC, Departamento de Engenharia Informática, Faculdade de Engenharia, Universidade do Porto, Portugal*

## HIGHLIGHTS

- A new dynamic resource management algorithm for concurrent workflows.
- Profit-aware algorithm that complies to users QoS requirements.
- Concurrent scheduling constrained to individual time and cost constraints.
- A realistic simulation that considers a bounded multi-port model.
- Results for randomly generated graphs as well as for real-world applications.

## ARTICLE INFO

*Article history:*
Available online 18 November 2016

*Keywords:*
Quality of service
On-line scheduling
Deadline
Budget

## ABSTRACT

The execution of a workflow application can result in an imbalanced workload among allocated processors, ultimately resulting in a waste of resources and a higher cost to the user. Here, we consider a dynamic resource management system in which processors are reserved not for a job but only to run a task, thus allowing a higher resource usage rate. This paper presents a scheduling algorithm that manages concurrent workflows in a dynamic environment in which jobs are submitted by users at any moment in time, on shared heterogeneous resources, and constrained to a specified budget and deadline for each job. Recent research attempted to propose dynamic strategies for concurrent workflows but only addressed fairness in resource sharing among applications while minimizing the execution time. The Multi-QoS Profit-Aware scheduling algorithm (MQ-PAS) proposed here is able to increase the profit achieved by the provider by considering the budget available for each job to define tasks priorities. We study the scalability of the algorithm with different types of workflows and infrastructures. The experimental results show that our strategy improves provider revenue significantly and obtains comparable successful rates of completed jobs.



## 1. Introduction

Many scheduling algorithms have been proposed to optimize the execution of a single workflow application on Heterogeneous Computing Systems (HCS) and with a single Quality of Service (QoS) parameter, usually minimizing the execution time of the application [1–3]. However, with the introduction of other factors such as execution cost, more objectives must be considered based on a user's QoS requirements. Conversely, other studies [4] show that the execution of a single workflow on a set of processors leads to a wastage of resources because the degree of parallelism of the application depends upon the workflow structure. Although the provider might charge for all processors during the execution time, doing so is evidence that resources are being wasted because workflow jobs are not able to use them fully.

Users tend to overestimate the resources that are required to guarantee that their jobs are not stopped before completing the required computations [5]. Therefore, a plausible solution is to run concurrent jobs in the same set of processors such that no static reservation is made per job. [6] showed the advantages of this approach, which led to a reduction of the number of nodes required and, consequently, to energy savings. To regulate the resource allocation procedure by user, the utility model [7] allows the provision of computing resources to consumers as needed and a payment model that charges for usage. This model has been used in computational grids and clouds. In this paper, we consider the utility model applied to a computational heterogeneous site.

DOI of original article: http://dx.doi.org/10.1016/j.future.2016.10.003.
* Corresponding author.
*E-mail addresses:* hamid.arabnejad@dcu.ie (H. Arabnejad), jbarbosa@fe.up.pt (J.G. Barbosa).

http://dx.doi.org/10.1016/j.future.2016.11.016
0167-739X/

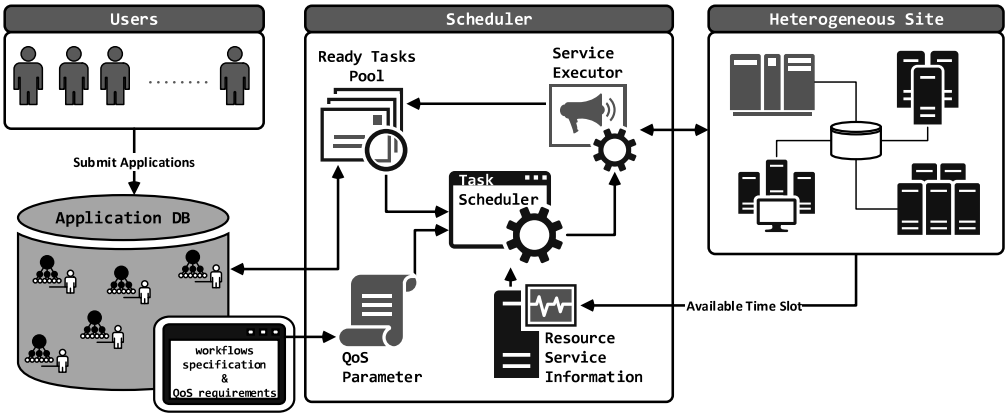


**Fig. 1.** A general view of the framework for concurrent workflow scheduling.

The algorithm proposed in this paper is intended to limit the costs for the user when running workflow applications and to allow the provider to obtain higher revenues by retaining a higher rate of successful completed applications and by prioritizing workflows attending to their available budget. In a real system, users submit their jobs at any moment in time. Therefore, the proposed algorithm is able to support the scheduling of concurrent workflow applications, which can be submitted at different moments in time and with individual QoS parameter values.

We introduce here a new scheduling strategy, Multi-QoS Profit-Aware scheduling algorithm (MQ-PAS), for scheduling concurrent workflow applications with multiple QoS constraints, here, time and cost. The MQ-PAS algorithm contains two main steps: first, it selects a task from each ready workflow and assigns a priority to each task based on the remaining time to application deadline and available budget. Second, for the higher priority task, MQ-PAS selects a suitable resource based on a quality measure computed for each resource.

The main contributions of this paper are (a) the proposal of a new low-time-complexity scheduling algorithm to address concurrent workflows constrained to time and cost. Our algorithm increases the provider profit while retaining, and in some cases increasing, the success rate of successful applications. (b) We present a realistic simulation that considers a bounded multi-port model in which bandwidth is shared by concurrent communications, and (c) we present results for randomly generated graphs and for real-world applications.

The remainder of the paper is organized as follows. In the next section, we describe a framework for concurrent job scheduling and the computational model. In Section 3, we describe the related work. That description is followed by the details of our MQ-PAS scheduling algorithm in Section 4. In Section 5, we show the benefits of the MQ-PAS via comparison and simulation. Finally, Section 6 presents conclusions and directions for future work.

## 2. Scheduling framework

The target utility computing platform is composed of a set of heterogeneous resources that provide services of different capabilities and costs [7]. Processor prices are defined such that the most powerful processor has the highest cost and the least powerful processor has the lowest cost. In a utility grid, the resource price is commonly defined and charged per time unit [8–10] such that if a task takes $k$ time units to process in a resource that costs $y$ euros per time unit, then the cost of executing the task in that resource is $k \times y$ euros.

Fig. 1 describes the framework modules required for an effective scheduling of concurrent workflows. Applications can be submitted into the system by any user and at any moment in time. The aim of this structure is to schedule tasks from the workflow applications into available resources, constrained by the user's QoS demands. Submitted applications are collected by an Application Data Base (DB) with user specifications and QoS requirements. In this architecture, a *Framework Scheduler* (FS) receives applications from the Application DB and generates task-to-resource maps for all applications. To make a proper decision concerning resource selection strategy for each application's task, FS needs the status information of available resources. The *Resource Service Information* is responsible for observing and collecting information about the current state of resources such as resource capacities, memory size, network bandwidth, availability, functionality and, in particular, the available time slots for processing tasks. In addition to resource status, the list of ready-to-execute tasks and user QoS requirements for each application are necessary to make a feasible schedule. The *Ready Task pool* module collects tasks that are ready to execute from among accepted workflow applications in the Application DB. A task is considered ready when its parents are executed. The *QoS Parameter* module contains the users' QoS requests for their workflow applications. These two modules are used to select the task and related application in each step of the scheduling process. The *Service Executor* module implements the task assignment by submitting each task to the selected resource and monitoring its execution. Finally, *Task scheduler* finds the suitable task-to-processor map for executing each ready task based on its QoS attributes and on the detailed information of each service.

### 2.1. Application model

A workflow application can be represented by a Directed Acyclic Graph (DAG) in which nodes represent tasks and edges represent task and data dependencies. A dependency ensures that a child node cannot be executed before all its parent tasks finish successfully and transfer the required child input data. The overall finish or completion time of an application is usually called the *schedule length* or *makespan*. The model to obtain the total execution cost for an application can consider computation costs, storage costs and data transfer costs.

A DAG can be modeled by a tuple, $G = \langle T, E \rangle$, consisting of a set of tasks, $T = \{t_1, t_2, \ldots, t_n\}$, in which $n$ is the number of tasks in the workflow, and a set of dependencies among the tasks, $E = \{\langle t_a, t_b \rangle, \ldots, \langle t_x, t_y \rangle\}$, in which $t_a$ and $t_x$ are parent tasks of $t_b$ and $t_y$, respectively. The $\overline{C}_{(t_i \to t_j)}$ represents the average communication time between the parent task $t_i$ and child task $t_j$, which is calculated based on the average bandwidth and latency among all processor pairs. This simplification is commonly considered to label the edges