Simulation Modelling Practice and Theory 58 (2015) 95–111

Contents lists available at ScienceDirect

Simulation Modelling Practice and Theory

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

Pareto tradeoff scheduling of workflows on federated commercial Clouds

Juan J. Durillo[a], Radu Prodan[a,*], Jorge G. Barbosa[b]

[a] Institute of Computer Science, University of Innsbruck, Austria
[b] Departamento de Engenharia Informática, Universidade do Porto, Portugal

A R T I C L E   I N F O

Article history:
Available online 31 July 2015

Keywords:
Scheduling
Cloud computing
Scientific workflows
Multi-objective optimisation
Pareto frontier
HEFT

A B S T R A C T

As distributed computing infrastructures become nowadays ever more complex and heterogeneous, scientists are confronted with multiple competing goals such as makespan in high-performance computing and economic cost in Clouds. Existing approaches typically aim at finding a single tradeoff solution by aggregating or constraining the objectives in an a-priory fashion, which negatively impacts the quality of the solutions. In contrast, Pareto-based approaches aiming to approximate the complete set of (nearly-) optimal tradeoff solutions have been scarcely studied. In this paper, we extend the popular Heterogeneous Earliest Finish Time (HEFT) workflow scheduling heuristic for dealing with multiple conflicting objectives and approximating the Pareto frontier optimal schedules. We evaluate our new algorithm for performance and cost tradeoff optimisation of synthetic and real-world applications in Distributed Computing Infrastructures (DCIs) and federated Clouds and compare it with a state-of-the-art meta-heuristic from the multi-objective optimisation community.

1. Introduction

Scientific workflows are a popular way of modelling applications to be executed on distributed computing infrastructures (DCI), such as Clouds. In this context, one of the most challenging tasks in the workflow lifecycle is how to schedule its tasks onto the heterogeneous available resources. Traditionally in parallel and distributed systems, workflow scheduling has been targeted to optimise the performance, measured in terms of makespan or the completion time of all tasks [1,2], which has been shown to be NP-complete.

In the context of Cloud computing, the user needs to care about an additional metric represented by the cost incurred by renting resources. Most of the commercial Clouds offer different types of resources at different prices. For example, in Amazon EC2[1] a user can choose among more than ten different types of resources, where the fastest is 35 times more expensive than the slowest.[2] Scheduling a workflow application becomes therefore a *multi-objective optimisation* problem [3] with at least two in-conflict criteria, *makespan* and *financial cost*, to which no single solution exists, but a set of tradeoff solutions called *Pareto front*.

* Corresponding author.
E-mail address: juan@dps.uibk.ac.at (J.J. Durillo).

[1] http://aws.amazon.com/ec2/pricing/.
[2] February 2015 prices for on-demand instances in US East zone.

http://dx.doi.org/10.1016/j.simpat.2015.07.001
1569-190X/© 2015 Elsevier B.V. All rights reserved.

Federated Clouds bring new opportunities when using commercial Clouds, as different providers may offer different services and resources with different performance and pricing models. In addition, a Cloud federation potentially increases the number and/or type of resources a user can use. A customer may be interested in scheduling low-priority tasks on the slow resources offered by a cheap provider, and high-priority critical tasks on the expensive fast resources offered by a high-performance provider, therefore keeping the multi-objective nature of the problem. Federated Clouds, however, pose additional drawbacks limiting the range of applications that can benefit from them. For example, resources belonging to different providers may be located in different areas connected via best-effort Internet, which is particularly problematic in the context of data-intensive applications. Companies may also impose restrictions over sensitive data that must stay within the fronts of a single institution, limiting optimisation opportunities. In this situations, it is not clear whether federated Clouds are an appealing alternative for workflow applications.

In this paper, we describe how the popular Heterogeneous Earliest Finish Time (HEFT) workflow scheduling algorithm can be extended to deal with multiple conflicting objectives when scheduling workflows on DCIs. The resulting algorithm, called *Multi-Objective HEFT (MOHEFT)* is able to deal with an arbitrary number of objectives, instantiated in this paper by makespan and financial cost. We evaluate MOHEFT in the context of a cloud federation, which also requires to customise it to properly deal with the peculiarities encountered today in public commercial Clouds, such as limited number of such as limited number of simultaneous resources and hourly billing intervals. In particular, we tackle the problem from the point of view of a broker coordinating the services of two federated Clouds providers with different performance and pricing models: Amazon EC2 and GoGrid. We analyse the performance and cost tradeoffs for scheduling workflows with different shapes (degree of parallelism), number of activities, and computation and communication requirements on DCIs and federated Clouds by comparing MOHEFT with a genetic state-of-the-art meta-heuristic from the multi-objective optimisation theory called SPEA2* [4].

The contributions of this paper are:

- We extended the HEFT algorithm for the multi-objective space;
- We customised the new algorithm to fit the characteristics of commercial Cloud federations, including maximum number of instances and hourly billing intervals;
- Our algorithm outperforms state-of-the-art heuristics from the multi-objective optimisation field such as SPEA2*;
- We analysed the benefits of using federated Clouds for different workflow types and found out that they can help in shortening their if transferring large amounts of data is not involved.

This paper extends our previous work [5] with additional experiments that evaluate the performance of MOHEFT when compared with HEFT and SPEA2* in terms of different types of workflow applications. Additionally, we include a new discussion that justifies the suitability of the MOHEFT algorithm for scheduling scientific workflow applications for the Cloud.

The paper is organised as follows. In the next section we describe the related work. Section 3 gives a short background on multi-criteria optimisation required for a better understanding of this work. Section 4 defines the abstract workflow, resource, and scheduling models underneath our approach. Section 5 presents the MOHEFT algorithm as an extension to the original HEFT heuristic, including a customisation for commercial Cloud infrastructures. We describe the experimental setup for validating our algorithm in Section 6, followed by the results in Section 7. Section 8 discusses the suitability of MOHEFT for scientific workflow applications and Section 9 concludes the paper.

2. Related work

In most related works, workflow scheduling optimising several competing objectives has been simplified either to a single-objective problem either by aggregating all the objectives in a single analytical function or by constraining the others. The main drawback of these approaches is that the aggregation and constraining are applied a priori, with a complete knowledge about the workflow, infrastructure, and in general about the problem being solved. Therefore, the computed solution may not properly capture the user preferences. On the other hand, few approaches that approximate the a Pareto set of tradeoff solutions have been proposed. Computing the complete Pareto front provides the user with a set of optimal solutions to select based on personal preferences and requirements, any may reveal solutions impossible to see beforehand.

The most common technique to combine multiple objectives into a single one is aggregation, i.e., to assign weights or preferences to the different objectives and to optimise the resulting aggregation function. The main differences among aggregation approaches is the way in which preferences are expressed. For example, [6] combines reliability (in terms of resource failures) and makespan using a weight vector. A similar approach has been used in [7,8], also for optimising reliability and time. A general disadvantage of these approaches is that the computed solution depends on the combination of the multiple objectives, which is made a priori and without a complete information about the problem being solved. This fact implies that, if the tradeoff or aggregation function does not capture the user preferences in an accurate way, the computed solution may not be satisfactory for the solved problem. Additionally, the objective functions require to be normalised to the same interval in order to properly capture these preferences.

Another way of combining the different objectives in a single objective function is by imposing user-defined constraints to each objective function (i.e. a desired value for each function to be optimised). Once the constraints have been set, the idea