J. Parallel Distrib. Comput. 74 (2014) 2152–2165

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

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

Multi-objective list scheduling of workflow applications in distributed computing infrastructures

Hamid Mohammadi Fard, Radu Prodan *, Thomas Fahringer

Institute for Computer Science, University of Innsbruck, Technikerstraße 21a, A-6020 Innsbruck, Austria

HIGHLIGHTS

- We propose a multi-objective scheduling framework for scientific workflows.
- We instantiate the framework for makespan, cost, energy, and reliability.
- We design a novel multi-objective list scheduling heuristic for workflows.
- We approximate the optimal solutions based on Pareto domination of user preferences.
- The solutions have better coverage compared to two related approaches.

ARTICLE INFO

Article history:
Received 16 June 2013
Received in revised form
30 October 2013
Accepted 5 December 2013
Available online 14 December 2013

Keywords:
Multi-objective scheduling
Scientific workflows
Distributed computing infrastructures

ABSTRACT

Executing large-scale applications in distributed computing infrastructures (DCI), for example modern Cloud environments, involves optimization of several conflicting objectives such as makespan, reliability, energy, or economic cost. Despite this trend, scheduling in heterogeneous DCIs has been traditionally approached as a single or bi-criteria optimization problem. In this paper, we propose a generic multi-objective optimization framework supported by a list scheduling heuristic for scientific workflows in heterogeneous DCIs. The algorithm approximates the optimal solution by considering user-specified constraints on objectives in a dual strategy: maximizing the distance to the user's constraints for dominant solutions and minimizing it otherwise. We instantiate the framework and algorithm for a four-objective case study comprising makespan, economic cost, energy consumption, and reliability as optimization goals. We implemented our method as part of the ASKALON environment (Fahringer et al., 2007) for Grid and Cloud computing and demonstrate through extensive real and synthetic simulation experiments that our algorithm outperforms related bi-criteria heuristics while meeting the user constraints most of the time.

1. Introduction

Scientific workflows emerged in the last decade as an attractive paradigm for programming large-scale applications in heterogeneous distributed computing infrastructures (DCI) such as Grids and Clouds. In this context, scheduling heterogeneous tasks including workflows is one of the traditional challenges in parallel and distributed computing. If we focus on the execution time also referred as makespan, the problem has been shown to be NP-complete, hence no polynomial algorithm for solving it exists (assuming $P \neq NP$). While this has been for decades the only optimization parameter of interest, modern DCIs are bringing along nowadays other parameters of equal importance such as reliability and economic cost (in Clouds), while recently energy consumption raises ever greater concerns too. Real-world scenarios are therefore confronted with a multi-objective optimization problem where many of these objectives are conflicting. For example, fast processors are typically rented by Cloud providers at higher prices, consume more energy, and may become unreliable due to the large user contention. In these scenarios, there is no single solution that optimizes all objectives, but a set of tradeoff solutions known as the Pareto frontier.

Until today, traditional scheduling researches [19] targeted makespan as the only optimization goal, while several isolated efforts addressed the problem by considering at most two objectives [17,21]. Although scheduling problems involve today multi-objective optimizations [14], a generic scheduling algorithm and framework for optimizing multiple conflicting objectives is still missing. Due to the NP-hard complexity of the makespan scheduling problem, practical approaches need to resort on heuristics to approximate the optimal solutions also in the multi-objective case. In this paper, we present a polynomial multi-objective algorithm for scientific workflow applications in heterogeneous DCIs that brings a two-fold novelty. First, we propose a general framework based on the multi-objective optimization theory for static

* Corresponding author.
E-mail addresses: hamid@dps.uibk.ac.at (H.M. Fard), radu@dps.uibk.ac.at, radu.prodan@uibk.ac.at (R. Prodan), tf@dps.uibk.ac.at (T. Fahringer).

0743-7315/$ – see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.jpdc.2013.12.004

scheduling of scientific workflows in DCIs. We analyze and classify different objectives with respect to their impact on the optimization process and present a four-objective case study comprising makespan, economic cost, energy consumption, and reliability. Second, we support our framework through a list scheduling heuristic algorithm capable of dealing with more than two objectives (as restricted by related works). The algorithm uses constraints specified by the user for each objective and approximates the optimal solution by applying a dual strategy: maximizing the distance to the constraint vector for dominant solutions and minimizing it otherwise.

The paper is organized as the follows. In Section 2 we review the related work, followed by a short background in multi-objective optimization theory in Section 3. In Section 4, we formalize the abstract application, objective, and platform models underneath our approach. In Section 5, we instantiate this model by a case study comprising makespan, cost, reliability and energy as objectives. Section 6 presents a new multi-objective list scheduling heuristic for workflow applications, illustrated through a small example in Section 7. We extensively evaluate our method in Section 8 for real-world and synthetic workflows and conclude in Section 9.

2. Related work

Most related works are bi-objective approaches which we organized in three categories: unconstrained, single-constraint, and Pareto-based.

2.1. Unconstrained approaches

The two workflow scheduling heuristics (list and genetic-based) proposed in [7] tradeoff execution time for reliability. The algorithms consider no constraint and no weight for the objectives, and only concentrate on fairly optimizing the objectives. Scheduling of pipeline workflows on homogeneous platforms with respect to latency and throughput has been studied in [3]. The algorithm does not consider general workflows and assumes one constrained objective in each execution. In [1], a list workflow scheduling heuristic that considers makespan as the first objective and reliability as the second one has been proposed. The key idea is to replicate the activities on proper resources to gain both reliability and makespan optimization. This lexicographic method considers no constraint for the objectives and also assumes that the objectives are arranged in the order of their importance. The generic multi-objective approach in [8] targets task scheduling of different users in two situations: constant and time-invariant penalty functions. Each user tries to increase an own utility, while the scheduler is responsible for increasing the overall utility of the users by assigning them priorities. The algorithm is restricted to independent tasks.

2.2. Single-constraint approaches

Two algorithms called LOSS and GAIN [17] schedule a directed acyclic graph (DAG) under a budget constraint based on a two-phase optimization process: the first phase optimizes one criterion, while in the second phase applies a budget constraint. In [22], the authors solve a bi-criteria workflow scheduling problem by minimizing the execution cost while meeting a deadline. In the first step, they divide the workflow deadline into sub-deadlines for all activities. In the second step, they model the sequential activities as a Markov decision process solved using a value iteration method. Both papers consider only one constrained objective and try to optimize the other with respect to the defined constraint. Furthermore, they use a rescheduling phase that introduces significant overhead that makes the scheduling process non-scalable.

The work in [21] proposes a bi-criteria genetic optimization algorithm that defines the fitness function as a combination of the partial fitness functions of the objectives. The algorithm is budget-constrained and considers no execution deadline. The algorithm in [2] targets DAG execution time minimization while keeping the number of failures equal to a constant x by replicating $x+1$ copies of each activity to different processors. The authors also suggest an extended algorithm which tries to improve the system reliability using redundancy.

2.3. Pareto-based approaches

The algorithm in [23] considers multi-objectives evolutionary algorithms for general workflow scheduling. The output is an approximation of the Pareto set and selecting the proper solution from this set remains a problem. The work in [11] is a similar approach for Grid workflow scheduling based on a multi-objective differential evolutionary algorithm that approximates the Pareto set by considering time and cost as objectives. In general, the major weakness of the evolutionary algorithms for scheduling problems is their slow convergence to good solutions, as we demonstrate in our experiments (Section 8). In contrast to Pareto-based approaches, our proposed algorithm can be categorized as an a-priori multi-objective scheduling method [14] looking for a single solution that satisfies the user's preferences and constraints. Our case study considers four objectives and uses Pareto relationships to find a solution that dominates or approaches the user constraints.

3. Multi-objective optimization background

We introduce several important concepts of the multi-objective optimization theory used in our method involving two steps: finding a set of optimal solutions and selecting the fittest solutions according to user preferences. We must distinguish two spaces as part of multi-objective optimization problems:

1. *solution* (or design) space X comprising all feasible solutions, for example the complete set of possible schedules of an application;
2. *objective* (or criterion) space O comprising an image of every element of X mapped to objective values.

In other words, each $x \in X$ maps to an image $o \in O$ which represents the objective values of x, as depicted in Fig. 1. A point $o \in O$ *dominates* $o' \in O$, denoted as $o \succ o'$, if o is not worse than o' with respect to all objectives and o is better for at least one of them. A point $o' \in O$ is said to be non-dominated if there is no $o \in O$ that dominates o'. A solution $x \in X$ is Pareto optimal (efficient) if its image in the objective space is non-dominated. The set of all Pareto-optimal solutions is called a *Pareto-optimal set*. The image set of all members of a Pareto-optimal set in the objective space is called a *Pareto frontier*. A vector comprising the best possible values for all objectives is called a *Utopia point* representing the ideal solution. Such a vector typically dominates the whole Pareto frontier and is therefore impossible to realistically achieve. A vector comprising the worst possible values for all existing objectives is called a *Nadir point*. The entire Pareto frontier dominates this point.

Let us assume that Fig. 1(b) depicts the objective space of the solution space of Fig. 1(a). The three solutions x, x' and x'' map to points o, o' and o'' in the objective space. If the optimization in this sample is minimizing both objectives O_1 and O_2, then the point o in the objective space is dominated by the point o'', while o'' has better values for both objectives O_1 and O_2. Two points o' and o'' are non-dominated points and consequently, the set $\{o', o''\}$ is the Pareto frontier. Therefore, the corresponding solutions x' and x'' are Pareto optimal and the set $\{x', x''\}$ is the Pareto set. The Utopia point is determined by selecting the best values of the objectives, as shown in Fig. 1(b). In contrast, the Nadir point is determined by selecting the worst values of the objectives. Typically, we need a single final solution selected by analysis of the preferences.