Future Generation Computer Systems ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

Execution time estimation for workflow scheduling

Artem M. Chirkin [a,b], Adam S.Z. Belloum [c], Sergey V. Kovalchuk [b,*], Marc X. Makkes [d], Mikhail A. Melnik [b,c], Alexander A. Visheratin [b], Denis A. Nasonov [b]

[a] ETH Zurich, Switzerland
[b] ITMO University, Russia
[c] University of Amsterdam, Netherlands
[d] VU University Amsterdam, Netherlands

HIGHLIGHTS

- An approach and algorithm to workflow execution time estimation is proposed.
- Formal description of the approach is provided.
- Workflow reduction techniques for the scheduling purposes are discussed.
- Scheduling GAHEFT algorithm with the use of the proposed algorithm is presented.
- Applicability of the proposed approach is demonstrated with experimental study.

ARTICLE INFO

Article history:
Received 17 May 2016
Received in revised form
30 September 2016
Accepted 7 January 2017
Available online xxxx

Keywords:
Workflow
Scheduling
Time estimation
Cloud computing
Genetic algorithm

ABSTRACT

Estimation of the execution time is an important part of the workflow scheduling problem. The aim of this paper is to highlight common problems in estimating the workflow execution time and propose a solution that takes into account the complexity and the stochastic aspects of the workflow components as well as their runtime. The solution proposed in this paper addresses the problems at different levels from a task to a workflow, including the error measurement and the theory behind the estimation algorithm. The proposed makespan estimation algorithm can be integrated easily into a wide class of schedulers as a separate module. We use a dual stochastic representation, characteristic/distribution function, in order to combine task estimates into the overall workflow makespan. Additionally, we propose the workflow reductions—operations on a workflow graph that do not decrease the accuracy of the estimates but simplify the graph structure, hence increasing the performance of the algorithm. Another very important feature of our work is that we integrate the described estimation schema into earlier developed scheduling algorithm GAHEFT and experimentally evaluate the performance of the enhanced solution in the real environment using the CLAVIRE platform.

1. Introduction

The execution time estimation is used mainly to support workflow scheduling. In this work we use "execution time", "runtime" and "makespan" as synonyms. In order to obtain the workflow runtime estimation, one needs to combine the execution time estimations of the tasks forming the workflow. Makespan estimation is an essential part of the scheduling optimization process because it greatly affects the quality of generated solutions no matter what optimization criteria are used. Accurate estimation is especially important for metaheuristic algorithms, which are based on estimations of evolving solutions. An incorrect estimation may lead to negative effects.

The runtime estimation problem is not as well-developed as the scheduling problem because several straightforward techniques exist, providing acceptable performance in many scheduling cases. However, these techniques may fail in some cases, especially in a deadline constrained setting. For example, consider a very basic approach when we take the average time of previous executions

* Corresponding author.
E-mail addresses: chirkinart@gmail.com (A.M. Chirkin), a.s.z.belloum@uva.nl (A.S.Z. Belloum), sergey.v.kovalchuk@gmail.com (S.V. Kovalchuk), M.X.Makkes@uva.nl (M.X. Makkes), mihail.melnik.ifmo@gmail.com (M.A. Melnik), alexvish91@gmail.com (A.A. Visheratin), denis.nasonov@gmail.com (D.A. Nasonov).

http://dx.doi.org/10.1016/j.future.2017.01.011
0167-739X/

of the same package as the only value representing the runtime of the task. If we apply it to a real case with a lot of tasks running in parallel, it will underestimate the total execution time, because it is very likely for at least one task to take more than average time for execution. In addition, when estimating the workflow execution time, we should take into account the dependency between the tasks and the heterogeneity of the tasks and computing resources. Another important feature of scientific workflows, from the estimation perspective, is that many components of the runtime are stochastic in their nature (data transfer time, overheads, etc.). These facets are not considered together in modern research; therefore, the problem requires a diligent investigation.

Besides scheduling, runtime estimation is used to give the expected price of the workflow execution when leasing Cloud computing resources [1]. Given the estimated time bounds, one can use a model described in [2] to provide a user with the pre-billing information.

The central idea of the work is to allow a scheduler to use the estimates as the random variables instead of keeping them constant, and provide the complete information about them (e.g. estimated distribution functions) at maximum precision. We compare our results to the only known-to-us alternative providing the distribution estimation – "fully probabilistic" algorithm – that is based on combining quantiles of the tasks' makespans. When calculating the workflow execution time, we assume that the tasks runtime models are adjusted to the distinct computing nodes; this allows us to separate the task-level (estimating the runtime on a given machine) and the workflow-level (aggregate the tasks' estimates into the overall makespan) problems. We also combine the proposed approach with the previously developed algorithm GAHEFT [3] and evaluate the performance of this schema in comparison with simple GAHEFT and a widely used heuristic algorithm called MinMin. The part of the paper devoted to makespan estimation is based on a Master's thesis by A. Chirkin [4]. See the thesis for the proofs of formulae and the implementation details.

As this work is dedicated to the estimation of schedules, proposed in this work scheduling algorithms are static. However, they might be included in our dynamic hybrid MHGH [3] scheme, which is based on GAHEFT. Therefore, by improving the static version of GAHEFT we also improve the dynamic version.

2. Related works

Workflow scheduling is known to be a complex problem; it requires considering various aspects of an application's execution (e.g. resource model, workflow model, scheduling criteria, etc.) [5]. Therefore, many authors describing the architecture of their scheduling systems do not focus on the runtime estimation. We separate the estimation from the scheduling so that the proposed approach (the workflow makespan estimation) can be used in a variety of scheduling algorithms, e.g. HEFT [6] and GAHEFT. Additionally, in order to monitor the workflow execution process or reschedule the remaining tasks, one can use the estimation system to calculate the remaining execution time at the moment when the workflow is being executed. Some examples of scheduling systems that can use any implementations of the runtime estimation module are CLAVIRE [7,8], a scheduling framework proposed by Ling Yuan et al. [9], and CLOUDRB by Somasundaram and Govindarajan [10].

There are no papers known to us, which are dedicated specifically to the workflow runtime estimation problem. Instead, there is a variety of scheduling papers that describe the runtime models in the context of their scheduling algorithms. We decided to classify the schedulers by the way they represent the execution time. Table 1 shows a classification of workflow scheduling algorithms;

Table 1
Usage of the runtime estimation in scheduling.

Time	Task	Workflow
Unspecified	Task ranking and makespans [11,12]	Architecture [9,10,13,14]
Ordinal	Round-Robin, greedy etc. [15,16]	–
Fixed	NP parallelizing models [17–19]	Optimization algorithms [11,15,20,21]
Stochastic	–	Chebyshev inequalities [22], composing quantiles [23]

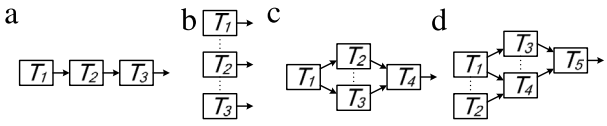

Fig. 1. Primitive workflow types: (a) sequential workflow—the workflow makespan is the sum of variables; (b) parallel workflow—the makespan is the maximum of variables; (c) complex workflow that can be split into parallel/sequential parts; (d) complex irreducible workflow.

the section below discusses the classes in details. Several papers are difficult to be classified because of two reasons: first, some approaches stay between two classes (e.g. in [11,12,17–19] the execution time is calculated as a mean of a random variable, but the variance of the time is not used); second, some researchers focus more on the architecture of scheduling software than on the algorithms, so one can use different estimate representations [9,10,14].

Ordinal time: The first class of schedulers uses task-level scheduling heuristics (which do not take into account the execution time of the workflow). An algorithm of such a scheduler uses a list of tasks ordered by their execution time. This ordering is the only information used to map the tasks onto the resources. First of all, this class is based on various greedy algorithms; two of them are described in [15]. Compared to the workflow-level algorithms, they are very fast but have been shown to be less effective [15]. A number of scheduling heuristics of this class is described in a paper by Gutierrez-Garcia and Sim [16].

Fixed time: The second class of schedulers considers the task runtime as a constant value. On the one hand, this assumption simplifies the estimation of the workflow execution time, because in this case there is no need to take into account a possibility that a task with the longer expected runtime finishes its execution faster than a task with the shorter expected time. Especially, this approach simplifies the calculation of the workflow execution time in case of parallel branches (Fig. 1(b)): if the execution time of branches is constant, then the expected execution time of the whole workflow is equal to the maximum of the branches' execution time. On the other hand, this assumption may lead to significant errors in the estimations and, consequently, to inefficient scheduling. In the case of a large number of parallel tasks, even a small uncertainty in the task execution time makes the expected makespan of the workflow longer. However, in some cases, a random part of the program execution time is small enough that this approach is used. One example of using fixed time is the work of M. Malawski et al. [24] who introduced a cost optimization model of workflows in IaaS (Infrastructure as a Service) clouds in a deadline-constrained setting.

Other examples of schedulers that do not exploit the stochastic nature of the runtime can be found in [11,15,20,21]. They calculate the workflow makespan; this means that they have to calculate execution time of parallel branches, but the stochastic and fixed approaches to estimate it are different. The main problem, again, is that the average of the maximum of multiple random variables is usually larger than the maximum of their averages.

Stochastic time: The last class of schedulers makes assumptions that have a potential to give the most accurate runtime predictions.

Please cite this article in press as: A.M. Chirkin, et al., Execution time estimation for workflow scheduling, Future Generation Computer Systems (2017), http://dx.doi.org/10.1016/j.future.2017.01.011