Reliable workflow scheduling with less resource redundancy ☆

Laiping Zhao [a,*], Yizhi Ren [b], Kouichi Sakurai [c]

[a] School of Computer Software, Tianjin University, Tianjin, China
[b] School of Software Engineering, Hangzhou Dianzi University, China
[c] Department of Informatics, Kyushu University, Fukuoka, Japan

ABSTRACT

We examine the problem of reliable workflow scheduling with less resource redundancy. As scheduling workflow applications in heterogeneous systems, either for optimizing the reliability or for minimizing the makespan, are NP-Complete problems, we alternatively find schedules for meeting specific reliability and deadline requirements. First, we analyze the reliability of a given schedule using two important definitions: *Accumulated Processor Reliability* (*APR*) and *Accumulated Communication Reliability* (*ACR*). Second, inspired by the reliability analysis, we present three scheduling algorithms: *RR* algorithm schedules least **R**esources to meet the **R**eliability requirement; *DRR* algorithm extends *RR* by further considering the **D**eadline requirement; and *dynamic* algorithm schedules tasks dynamically: It avoids the "Chain effect" caused by uncertainties on the task execution time estimates, and relieves the impact from the inaccuracy on failure estimation. Finally, the empirical evaluation shows that our algorithms can save a significant amount of computation and communication resources when performing a similar reliability compared to Fault-Tolerant-Scheduling-Algorithm (FTSA) algorithm.

© 2013 Elsevier B.V. All rights reserved.

1. Introduction

With the new deployed machines and old, slow machines are replaced with new, fast ones continuously, distributed systems (e.g., Cloud, Grid) are believed to become more heterogeneous in the future. Resource management is a big challenge in large-scale heterogeneous systems due to the various configurations or capacities of the hardware or software. In particular, efficient scheduling algorithms play a critical role to attain the requirements of users or systems.

Workflow scheduling aims to find a mapping of the tasks to the processors, so that specific requirements are satisfied while respecting task precedence constraints. For example, HEFT and CPOP [34] algorithms sort all tasks into order based on the task precedence constraints, then dispatch them to different processors with the objective of reducing the scheduling length. Although algorithms have been proposed to reduce the schedule length [2,8,34], minimizing the schedule length has been proven to be NP-Complete [16]. That is, there exists no polynomial-time algorithm (unless P = NP) to find a schedule with minimum schedule length. Therefore, we alternatively try to find a schedule that is capable of meeting a predefined deadline rather than minimizing the execution time.

Reliability has become another very important issue in workflow scheduling since the work of [31], and a number of algorithms have been proposed to improve the reliability thereafter [14,15,19,20,32]. In order to provide high reliability, active replication scheme and backup/restart scheme [9], corresponding to resource and time redundancy respectively, are widely applied in scheduling. For the active replication scheme, each task is replicated on several processors simultaneously, and the task will succeed if at least one of them does not fail [6,7,17,18]. For the backup/restart scheme, whenever a processor fails, the task will be rescheduled on a backup processor to proceed [41] [42]. Checkpoint/restart scheme can be considered as an improved version of the backup/restart scheme. That is, when a failure occurs, the task can be restarted from the latest checkpoint, not from the very beginning. This could reduce the resources enrolled for execution. However, both backup/restart scheme and checkpoint/restart postpone the job execution, which probably results in violation of deadline. Moreover, since randomized numbers may lead to total different results, problems with the backup/restart scheme become even more complex when a randomized number is used. Therefore, they are supposed to be complementary to active replication and do not exclude each other.

Scheduling with replications for optimizing reliability on heterogeneous platforms has been addressed in [5], and unfortunately proven to be NP-Complete too. That is, there exists no polynomial-time scheduling algorithm that is able to maximize the reliability (unless P = NP). To this end, we here again choose an alternative way for reliable workflow scheduling: schedule task replicas to processors with the objective of meeting a reliability requirement rather than maximizing the reliability.

The drawback of scheduling with replications is obvious: it consumes a vast amount of resource for fault tolerance, and many of them are just a waste of resources. For example, in order to tolerate two failures, every task of a workflow has to be replicated for at least three times. In case of a workflow consisting of thousands of tasks, many of these copies may be just a waste of resources because a task will succeed as long as one of them does not fail. Therefore, to improve the efficiency, we focus on the problem of reliable workflow scheduling with less resource redundancy in this paper. More precisely, given a workflow represented job, and the probability of failure for a task on a processor is known, then tasks of the job will be replicated for several times and dispatched onto corresponding processors. In this process, we are required to minimize the number of replicas for meeting the reliability requirement, and the final schedule should not violate the deadline.

The first arisen challenge is on reliability analysis. Benoit et al. [4] proved that, evaluating the reliability of general schedules, no matter the failures follow a transient or a fail-stop manner, belongs to the #P'-Complete problems, which are at least as hard as NP-Complete problems. Hence, even obtaining a schedule for the workflow, we are still unable to compute its reliability in polynomial time, thereby unable to verify whether a schedule satisfies the reliability requirement or not. The complexity of evaluating the reliability of a schedule following a strict/fail-stop manner is not yet proven, and is still an open problem. We regretfully are unable to prove the complexity of evaluating the reliability of a strict/fail-stop schedule. However, this does not interfere with our evaluation of algorithms, because we provide an analysis on the upper and lower bound of the reliability.

The second challenge is how to compute the number of replications for each task. Replications employed by each task should be minimized for reducing resource usage, yet cannot violate the overall reliability requirement. Fortunately, the analysis on the lower bound makes this problem significantly simpler, and directly inspires us with the design of RR algorithm.

In accordance with these challenges, our contributions include:

- Given a schedule, we can compute the upper bound and lower bound of the reliability of the schedule. The lower bound can be used for verifying the reliability satisfaction, while the upper bound can be used for reliability comparison between our proposal and existing algorithms, e.g., FTSA [6].
- The computation process on the lower bound inspires us with a algorithm for satisfying reliability requirement, i.e., RR algorithm (Two *"R"*s are for *"Reliability"* and *"Resources"*.), which greedily schedules processors according to their accumulated reliability. DRR algorithm extends RR algorithm regarding both reliability and deadline requirements.
- We further consider the limit that lacking knowledge of reliability in practical systems, and propose a dynamic scheduling algorithm, which makes more efficient utilization of available resources while satisfying the reliability and deadline requirements.
- We experimentally evaluate the efficiency of the algorithms. With setting the reliability requirement equal to the upper bound of schedules produced by FTSA and FTSA_BL [6], our algorithms can satisfy the reliability and deadline requirements but with a significant reduction on replications.

The rest of this paper is organized as follows. In Section 2, we present the system model and formally define the problem. Section 3 presents our analysis on reliability of a given schedule. And inspired by the analysis, we outline our two reliable workflow scheduling algorithms: RR and DRR in Section 4. We further propose a dynamic algorithm in Section 5. And the experimental evaluation is presented in Section 6. We then introduce the related works in Section 7, and briefly discuss the proposed algorithms in Section 8. Section 9 presents the conclusion and future works.