ARTICLE IN PRESS

Computer Networks xxx (2015) xxx–xxx

Contents lists available at ScienceDirect

Computer Networks

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

Multihybrid job scheduling for fault-tolerant distributed computing in policy-constrained resource networks

Yong-Hyuk Moon [a,*], Chan-Hyun Youn [b]

[a] Electronics and Telecommunications Research Institute, South Korea
[b] Department of Electrical Engineering, Korea Advanced Institute of Science and Technology, South Korea

ARTICLE INFO

Article history:
Received 17 June 2014
Received in revised form 1 December 2014
Accepted 4 February 2015
Available online xxxx

Keywords:
Job scheduling
Fault tolerance
Policy heterogeneity
Multiobjective optimization
Distributed computing
Genetic algorithm

ABSTRACT

Unpredictable fluctuations in resource availability often lead to rescheduling decisions that sacrifice a success rate of job completion in batch job scheduling. To overcome this limitation, we consider the problem of assigning a set of sequential batch jobs with demands to a set of resources with constraints such as heterogeneous rescheduling policies and capabilities. The ultimate goal is to find an optimal allocation such that performance benefits in terms of makespan and utilization are maximized according to the principle of Pareto optimality, while maintaining the job failure rate close to an acceptably low bound. To this end, we formulate a multihybrid policy decision problem (MPDP) on the primary-backup fault tolerance model and theoretically show its NP-completeness. The main contribution is to prove that our multihybrid job scheduling (MJS) scheme confidently guarantees the fault-tolerant performance by adaptively combining jobs and resources with different rescheduling policies in MPDP. Furthermore, we demonstrate that the proposed MJS scheme outperforms the five rescheduling heuristics in solution quality, searching adaptability and time efficiency by conducting a set of extensive simulations under various scheduling conditions.

1. Introduction

One of the most challenging requirements of job scheduling systems currently being developed is ensuring that they elastically allocate computing resources to jobs despite the unpredictable occurrence of resource failures. With increased complexity, dynamic features, and various uncertainties, resource failure [1–3] is the rule rather than the exception in distributed computing systems (DCS) such as grid, peer-to-peer, and cloud computing. It has been reported that over 75% and 70% of the resources have failure rates of about 20% and 40% in workload archives such as DEUG, and UCB and SDSC [3], respectively. From these application-level traces, we found that most resources have relatively high failure probabilities. It is also recognized that failures can significantly affect scheduling performance and that the large number of job failures is still caused by resource fluctuations and unavailability, as discussed in [2]. Specifically, Ref. [4] has exhibited the statistical overviews on job execution in ten different DCS in which the proportion of failed jobs varied from 1.23% to 83.94% and failed jobs consume lots of computational power of resources, ranging from 0.41% to 73.88%. Although most resources are managed by autonomous domains (ADs), each AD tends to employ proprietary administrative rules (i.e., rescheduling policies) for job recovery. This fact further complicates the large scale resource pooling for reliable job execution in a cooperative manner. As a result, in addition to extending known

* Corresponding author.
E-mail addresses: yhmoon@etri.re.kr (Y.-H. Moon), chyoun@kaist.ac.kr (C.-H. Youn).

http://dx.doi.org/10.1016/j.comnet.2015.02.030
1389-1286/

rescheduling policies, it requires a new scheduling algorithm that can efficiently guarantee fault tolerance.

There are some scheduling approaches recently studied in order to alleviate the undesirable effects of resource failures with different policies. In [5], a task re-execution policy employed in Hadoop has been currently studied in order to analyze its impact on job completion reliability and time from a theoretical viewpoint. However, it is not clear which configuration of this policy is sufficient to prevent a running job from being interrupted by resource failures. To overcome this limitation, Zhang et al. [6] have integrated backfilling and migration with well-established gang scheduling strategy. The fully integrated allocation method consistently outperforms the others. Nevertheless, the authors have not explicitly addressed what specific conditions can decide the most suitable combination of different strategies. For similar reason, a prior study [7] proposes a concept of dynamic policy switching at run time for interactive jobs, which require immediate execution. Its weakness is that the proposed scheme is irrelevant for batch jobs that are submitted to a job-ready queue and await their turn to run. On the contrary to that work, in [8] the authors propose how to decide a near-optimal size of hybrid cluster in clouds for accelerating batch analytics. From this study, we have found that it is possible to tolerate a high degree of instability in clouds if system-level metrics are only given; however, the degree of performance gain is not strongly correlated with the resource usage cost. To tackle this problem, Farahabady el al. [9] proposes a new scheduling algorithm which generates solutions with the Pareto optimality between cost and performance. The key question of how to dynamically cope with performance variability of underlying resources due to failures remains unanswered. Furthermore, the performance impacts of different combinations of job selection policies and job dispatching mechanisms have been presented for the Bag-of-Tasks (BoT) allocation [10]. Although this study has shown monotonic improvement in some cases, they have only emphasized the static combinations of different job allocation strategies and have paid less attention to the fact that resources are controlled by heterogeneous rules. In [11], the authors have analyzed the correlation between job sizes and scheduling algorithms. However, they have not considered the potential gains of using different scheduling approaches simultaneously. Besides, the failure condition has not been considered a major issue.

To the best of our knowledge, there have not been direct contributions to tackle the policy heterogeneity problem in the context of job scheduling. Thus, a new policy-integrated scheduling scheme needs to answer the following research questions, which have remained unsolved: (1) How much performance improvement is expected, as compared to static rescheduling policies? (2) How much fault tolerance can it provide without sacrificing the main objectives considering a large fraction of resources in availability? (3) Assuming that different policies are supported in various ADs, how can it adapt to this situation regarding scheduling quality and complexity? In this paper, we propose a multihybrid job scheduling (MJS) scheme in order to deliver robust resource allocation for computational batch jobs under heterogeneous policies. The proposed MJS scheme adopts a messy genetic algorithm (mGA) [12], which has been applied to solve combinatorial optimization problems. The main contributions of this study are as follows:

- We formulate the aforementioned problem as a multihybrid policy decision problem (MPDP) on the primary-backup fault tolerance model and discuss how an optimal solution can be quantified in terms of scheduling quality.
- To solve MPDP effectively, we propose a new MJS scheme that finds an optimal schedule even in a large search space by using stochastic search operations of mGA within relatively low and acceptable complexity.
- The mGA-based MJS scheme demonstrates high fault tolerance without sacrificing the other objectives, such as makespan and load balance, unlike the static approaches and deterministic algorithms (e.g., *minmin* [13]), even in the policy-constrained DCS.

The remainder of this paper is organized as follows. Section 2 presents a fault-tolerant job scheduling model with the four rescheduling policies. We formulate MPDP in Section 3 and then propose an mGA-based MJS scheme, focusing on solution representation and realization in Section 4. Section 5 provides analytical views on the terms of approximation of computational complexity and possibility of convergence of the proposed scheme. Simulation results and discussions are in Section 6. Finally, we conclude the paper in Section 7.

2. Fault-tolerant job scheduling model

A job scheduling system in DCS consists of three tiers: users submitting batch jobs $\mathcal{J} = \{j_i | i = 1, 2, 3, \ldots c, N\}$, a job scheduler, and computing resources $\mathcal{R} = \{r_x | x = 1, 2, 3, \ldots, M\}$. Each resource x, r_x, has its relative computational speed c_x, which is assumed to be an arbitrary and non-decreasing (i.e., concave) function over time. A job scheduler distributes stochastic workloads $\nu(\Delta t)$ arriving in a given period of time, Δt across allocated resources. An arriving j_i can be distinguished by its arrival time $\tilde{a}_i$, job length l_i, and deadline $\tilde{d}_i$. $\tilde{a}_i$ should be bounded to $[0, \Delta t]$ and l_i indicates the time spent by r_x with $c_x = 1$ (i.e., reference computing element) to execute j_i. Therefore, the effective length of j_i can be calculated as $\tilde{l}_{i,x} = l_i / c_x$, if there is no failure. We suppose that at time zero, all resources are operable and jobs can be allocated to them. Resources can execute, at most, a single job at a time and can also fail at any random time. Table 1 summarizes a number of commonly used symbols and we will discuss some of them further after that.

2.1. Application workload

Realistic workloads comprise mostly single-process jobs, and 85–95% of them are batch jobs as measured in the several grid workloads [14]. Likewise, we adopt the

Please cite this article in press as: Y.-H. Moon, C.-H. Youn, Multihybrid job scheduling for fault-tolerant distributed computing in policy-constrained resource networks, Comput. Netw. (2015), http://dx.doi.org/10.1016/j.comnet.2015.02.030