J. Parallel Distrib. Comput. 74 (2014) 2180–2192

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

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

Competitive online adaptive scheduling for sets of parallel jobs with fairness and efficiency☆

Hongyang Sun [a,*], Wen-Jing Hsu [a], Yangjie Cao [b]

[a] School of Computer Engineering, Nanyang Technological University, Singapore
[b] School of Software Engineering, Zhengzhou University, China

HIGHLIGHTS

- We study online adaptive scheduling for sets of parallel jobs on multiprocessors.
- We propose an improved algorithm with both fairness and efficiency.
- The proposed algorithm is asymptotically competitive for set response time.
- We provide a framework for analyzing a family of algorithms with provable efficiency.
- We consider hierarchical scheduling and present a fair and efficient solution.

ARTICLE INFO

Article history:
Received 21 February 2013
Received in revised form
6 November 2013
Accepted 5 December 2013
Available online 14 December 2013

Keywords:
Adaptive scheduling
Parallel jobs
Set response time
Multiprocessors
Online algorithm
Competitive analysis
Hierarchical scheduling
Fairness
Efficiency

ABSTRACT

We study online adaptive scheduling for multiple sets of parallel jobs, where each set may contain one or more jobs with time-varying parallelism. This two-level scheduling scenario arises naturally when multiple parallel applications are submitted by different users or user groups in large parallel systems, where both user-level fairness and system-wide efficiency are of important concerns. To achieve fairness, we use the well-known equi-partitioning algorithm to distribute the available processors among the active job sets at any time. For efficiency, we apply a feedback-driven adaptive scheduler that periodically adjusts the processor allocations within each set by consciously exploiting the jobs' execution history. We show that our algorithm achieves asymptotically competitive performance with respect to the set response time, which incorporates two widely used performance metrics, namely, total response time and makespan, as special cases. Both theoretical analysis and simulation results demonstrate that our algorithm improves upon an existing scheduler that provides only fairness but lacks efficiency. Furthermore, we provide a generalized framework for analyzing a family of scheduling algorithms based on feedback-driven policies with provable efficiency. Finally, we consider an extended multi-level hierarchical scheduling model and present a fair and efficient solution that effectively reduces the problem to the two-level model.

1. Introduction

1.1. Background

Scheduling parallel jobs on multiprocessor systems has been a fundamental area of research in computer science for decades [10,14,18]. As parallel systems are increasingly deployed to provide high-performance computing services, such as in cloud computing and the large-scale data centers, efficient scheduling on these platforms will play a more important role in boosting application performance and increasing system utilization.

Most parallel systems today are shared by multiple users, and a common scenario arises when each user submits several jobs to the system. A natural objective in this case is to achieve efficient execution of the jobs while at the same time offering a level of fairness among different users. In this paper, we consider such a scenario, in which a collection of parallel job sets needs to be scheduled on a multiprocessor system and each job set corresponds to the set of applications submitted by a particular user or user group. We are interested in the *response time* of a job set, which is the duration between when the job set is submitted and when all jobs in the job set are completed. The objective is to minimize the sum of the response times of all job sets, or the *set response time*. As pointed out by Robert and Schabanel [19], the metric of set response time benchmarks both fairness and efficiency of a scheduling algorithm.

☆ A preliminary version of this paper has appeared in the Proceedings of the 17th IEEE International Conference on Parallel and Distributed Systems, December 2011.

* Corresponding author.
E-mail addresses: sunh0007@ntu.edu.sg (H. Sun), hsu@ntu.edu.sg (W.-J. Hsu), caoyj@zzu.edu.cn (Y. Cao).

0743-7315/$ – see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.jpdc.2013.12.003

In fact, it represents a more general performance measure that incorporates two widely used metrics, namely, *total response time* and *makespan*, as special cases. Suppose that each job set in the collection contains only a single job, the set response time becomes the total response time of all jobs in the collection. At the other extreme, if the collection contains only a single job set, the set response time is simply the makespan of all jobs. To schedule a collection of job sets, an algorithm needs to allocate processors at two separate levels, namely, the *job-set level* and the *job level*. In particular, it needs to first specify the number of processors allocated to each job set, and then decides the processor allocation for the jobs within each job set. Such a two-level scheduling model is considerably more challenging compared to the traditional single-level scheduling [8,7,1,11,25], where an algorithm only needs to decide the processor allocation for a flat collection of jobs.

We consider parallel jobs with time-varying parallelism profile, which is commonly observed in many applications that go through different phases in their executions. Each phase of a job is specified by an amount of work to be done and a speedup function, which we assume in this paper is linear up to a phase-dependent maximum. (See Section 2.1 for the detailed job model.) Moreover, the parallel jobs are assumed to be *malleable* [10] in nature, that is, they can adjust to the changing processor allocations at runtime. Such malleability is enabled by the more flexible runtime systems [30,21,6,23] that have emerged in the last decade as well as the state-of-the-art virtual machine (VM) technology [2,16,22] that makes adaptive scheduling possible with little or negligible overhead. In contrast to *static scheduling* [14], which allocates a fixed set of processors to a job throughout its execution, *adaptive scheduling* can benefit from the time-varying characteristic of the jobs' resource requirements and hence appears to be a more promising approach to scheduling jobs in modern parallel systems. We adopt the *online non-clairvoyant* scheduling model [17,11], which requires an algorithm to make all scheduling decisions in an online manner without any knowledge of the jobs' future characteristics, such as their release time, remaining work and parallelism profile. This is a natural assumption since such information is generally not available to the operating system schedulers. We measure the performance of an online adaptive scheduler using the standard *competitive analysis* [3], which compares its set response time with that of an optimal offline scheduler.

1.2. Related work

A well-known online adaptive scheduler is Equi-partitioning (EQUI) [29], which at any time divides the total number of processors evenly among all active jobs in the system. This algorithm, although simple, is able to ensure fairness by automatically adjusting the processor allocations whenever a new job is admitted into the system or an existing job is completed and leaves the system. In fact, such a simple notion of fairness is sufficient to guarantee satisfying performance when each user submits only one job. In particular, Edmonds et al. [8] showed that EQUI is $(2+\sqrt{3})$-competitive with respect to the total response time of all jobs if they are released at the same time. Using *resource augmentation analysis* [13], Edmonds [7] also showed that EQUI is $(2+4/\epsilon)$-competitive for arbitrary released jobs with processors whose speed is $2+\epsilon$ times faster than the optimal, for any $\epsilon > 0$.[1] However, despite its good performance for the total response time, EQUI fares poorly in terms of the makespan, which to a certain extent reflects the system efficiency when there is only one user in the system. Since EQUI does not consider how efficiently each job is able to utilize the allocated processors, it may under-utilize the resources especially when different jobs can have very different processor requirements. Specifically, Robert and Schabanel [19] showed that EQUI is $\Theta(\frac{\ln n}{\ln\ln n})$-competitive with respect to the makespan even if all jobs are batch released, where n is the total number of jobs submitted to the system.

It turns out that both user-level *fairness* and system-wide *efficiency* are critical when minimizing the set response time for a collection of job sets. In [19], Robert and Schabanel applied EQUI to both levels by equally dividing the total number of processors among the active job sets and within each job set equally dividing the allocated processors among its active jobs. They showed that the resulting algorithm EQUI∘EQUI has a competitive ratio of $(2+\sqrt{3}+o(1))\frac{\ln n}{\ln\ln n}$ with respect to the set response time when all job sets are batch released, where n is the maximum number of jobs in any job set. The result suggests that the set response time ratio of a scheduling algorithm actually encompasses the total response time ratio and the makespan ratio of the corresponding algorithms at the job-set level and the job level, respectively. Hence, it is important to retain both fairness and efficiency in order to achieve satisfying performance for this general scheduling metric.

To improve the system efficiency, feedback-driven adaptive schedulers [1,11,25] were recently proposed. Unlike EQUI, which obliviously allocates processors to the jobs regardless of their actual resource requirements, feedback-driven algorithms periodically adjust processors among the jobs by consciously exploiting the jobs' execution history. In particular, Agrawal et al. [1] introduced the A-GREEDY scheduler that periodically collects the resource utilization of each job, and based on this information estimates the job's future processor requirement. It has been shown that A-GREEDY wastes at most a constant fraction of a job's allocated processors, and thus achieves efficient processor utilization [1]. Furthermore, by combining A-GREEDY with a conservative resource allocator, such as Dynamic Equi-partitioning (DEQ) [15], He et al. [11] showed that the feedback-driven algorithm AGDEQ is asymptotically $O(1)$-competitive with respect to the makespan regardless of the number of jobs in the system. Recently, Sun et al. [25] proposed another feedback-driven adaptive scheduler ACDEQ, which uses a control-theoretic approach to estimate the jobs' processor requirements and it has been shown to have better feedback stability and system efficiency.

1.3. Our contributions

Aiming at both user-level fairness and system-wide efficiency, we bring together the benefit of the EQUI algorithm [29] and that of the feedback-driven scheduler AGDEQ [11], and propose a both fair and efficient online adaptive algorithm EQUI∘AGDEQ for scheduling any collection of job sets. We show that the set response time ratio of our algorithm indeed combines the total response time and the makespan ratios of the respective algorithms in a non-trivial manner. Our first contribution is to bound the asymptotic competitive ratio (see Section 2.2 for the detailed definition) of EQUI∘AGDEQ, which is summarized in the following.

- EQUI∘AGDEQ is $O(1)$-competitive in the asymptotic sense with respect to the set response time when all job sets are batch released. The exact ratio depends on the constant parameters of the AGDEQ algorithm, and it is formally stated in Theorem 1 (Section 4.2). This result improves the competitive ratio of $\Theta(\frac{\ln n}{\ln\ln n})$ achieved by the EQUI∘EQUI algorithm [19] for sufficiently large jobs, where n is the maximum number of jobs in any job set. The improvement is a direct result that EQUI∘AGDEQ exhibits both fairness and efficiency while EQUI∘EQUI provides only fairness but lacks efficiency.

[1] Edmonds and Pruhs [9] recently proposed a variant of the EQUI scheduler, called Latest Arrival Processor Sharing (LAPS), which at any time shares the total number of processors evenly among the β fraction of the active jobs with the latest release time, for any $0 < \beta \leq 1$. They showed that by varying the parameter β, LAPS provides a tradeoff between the augmented processor speed $s = 1+\beta+\epsilon$ and the competitive ratio $4s/(\beta\epsilon)$, for any $\epsilon > 0$.