Scalable hierarchical scheduling for multiprocessor systems using adaptive feedback-driven policies

Yangjie Cao
Dept. of Electronic and Information Engineering
Xi'an Jiaotong University
Xi'an, China
Email: caoyj@stu.xjtu.edu.cn

Hongyang Sun
Dept. of Computer Engineering
Nanyang Technological University
Singapore
E-mail: sunh0007@ntu.edu.sg

Depei Qian
Dept. of Electronic and Information Engineering
Xi'an Jiaotong University
Xi'an, China
Email: depeiq@xjtu.edu.cn

Weiguo Wu
Dept. of Electronic and Information Engineering
Xi'an Jiaotong University
Xi'an, China
Email: wgwu@xjtu.edu.cn

***Abstract*—This work addresses the problem of allocating resource-intensive parallel jobs on multicore- and multiprocessor-based systems, where the performance gains largely depend on effectively exploiting application parallelization across the available parallel computing resources. The objective is to find efficient allocation approaches that minimize the parallel jobs' completion time, i.e. makespan. Integrating feedback-driven adaptive strategies, we present a general hierarchical scheduling framework and show that two hierarchical scheduling algorithms: ABG-DS and AG-DS achieve scalable performance in term of makespan regardless of the number of hierarchical levels. Specifically, we prove that both ABG-DS and AG-DS have O(1)-competitive ratio for batched parallel jobs. Extending an existing tool, called Malleable-Lab, we evaluate the performance and scalability of our proposed algorithms and compare with that of well-known EQUI-based strategies. The simulation results demonstrate that both ABG-DS and AG-DS generally outperforms EQUI-EQUI for a wide range of parallel workloads. Moreover, feedback-driven adaptive scheduling algorithms show better scalability when the number of levels increases in the scheduling hierarchy.**

I. Introduction

Multicore and multiprocessor computers are increasingly used to support a wide range of parallel and distributed computing environments, such as multiclusters, Grid and more recently Cloud computing infrastructure. In these environments, the productivity and performance gains largely depend on effectively exploiting application parallelization across the available parallel computing resources. With the rapidly evolving nature of parallel programming paradigms, however, the recent parallel workloads with a wide spectrum of parallelism and various QoS requirements poses an even harder challenge to the scheduling community.

In this paper, we study the adaptive scheduling for malleable parallel jobs on multiprocessor-based systems. To handle complicated resource allocation and aggregate performance requirements, a hierarchical scheduling framework is applied in our study. Hierarchical scheduling provides a more flexible way to separate the concerns of scheduling at different levels and has drawn much attention in literature [1]–[3]. In a hierarchical scheduling framework, proportional-share scheduling algorithms are commonly used to proportionally share all available resources among competing requests. As illustrated in [2], traditional proportional-share scheduling algorithms, which are designed for uniprocessor environment, can result in unbounded unfairness and starvation when employed in multiprocessor environments. Using adaptive feedback-driven strategies, we focus on studying online scheduling to achieve both fairness and efficiency for executing parallel applications on hierarchical systems. The objective is to minimize the makespan, i.e., the completion time of the last completed job in the job set. We adopt the online non-clairvoyant scheduling model, which requires the algorithm to operate in an online manner, that is, to make irrevocable decisions in response to each incoming request with no knowledge of the jobs' future characteristics, such as their internal parallelism, and remaining work. To measure the performance of an online scheduling algorithm, we employ the competitive analysis [4], which compares the performance of an online algorithm with that of an optimal offline scheduler.

Non-clairvoyant scheduling was introduced by Motwani et al. [5] in an attempt to design algorithms that are provably efficient for practical purposes. To date, several extensions have been made to take advantage of time- or space-shared resources. In the multiprocessor environments, the well-known non-clairvoyant scheduling algorithm is EQUI (Equi-partition) [6], [7], which equally shares available computing power among all jobs. For makespan minimization Robert et al. [8] show that EQUI achieves competitive ratio of $O(\frac{\ln n}{\ln \ln n})$ and that no better ratio is possible. The closely related work to our study in the similar setting is the work done by Robert et al. [8] and Sun et al. [9]. In [8], the authors organize the jobs into different sets and present an online scheduling algorithm: EQUI∘EQUI, which consists in 1) splitting evenly the available processors among the sets and then 2)

978-0-7695-4190-7/10 $26.00 © 2010 IEEE
DOI 10.1109/ISPA.2010.68

splitting evenly these processors among the jobs of each set. They consider the metric of setflowtime, i.e. the total makespan of all sets, and proved that EQUI∘EQUI achieved a competitive ratio $(2+\sqrt{3}+o(1))\frac{\ln n}{\ln \ln n}$. In [9], Sun et al. considered the same metric but they combined EQUI with feedback-driven adaptive policies ABG [10] and AG [11] to propose EQUI∘ABG and EQUI∘AG. They showed that both EQUI∘ABG and EQUI∘AG achieved O(1)-competitiveness with respect to the setflowtime for batched jobs.

In this paper, we present a hierarchical scheduling framework and integrate the feedback-driven adaptive strategies to optimize overall resource allotment. Specifically, we use feedback-driven schedulers: ABG and AG to indicate processor requirements for the jobs at the bottom level and at higher levels we present a scheme, called DS (Desire-Sum) for aggregating the resources requirements at each intermediate nodes. We show that both ABG-DS and AG-DS achieve scalable performance in terms of makespan by proving that their competitive ratios are identical to those in the two-level setting [10], [12], namely O(1)-competitive for batched jobs. To evaluate the performance of hierarchical scheduling algorithms, we extend an existing tool, called Malleable-Lab [13], to support hierarchical scheduling and implement three algorithms: ABG-DS, AG-DS, and EQUI-EQUI which is a variation of EQUI for hierarchical scheduling environments. The simulation results demonstrate that both ABG-DS and AG-DS generally achieve better performance than EQUI-EQUI with respect to makespan. Moreover, ABG-DS has shown better scalability than AG-DS, which leads to ABG-DS having much better and more stable performance in simulation.

The rest of this paper is organized as follows. In Section 2, we formally define the hierarchical scheduling model. Section 3 describes the hierarchical algorithms ABG-DS and AG-DS and analyzes the performance of these algorithms with respect to makespan. Section 4 presents the simulation and its results. Finally, Section 5 concludes the paper with future directions.

II. Hierarchical Scheduling Model

We introduce the hierarchical scheduling model in detail in this section. There are a set of n jobs, $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$, to be scheduled on a tree structure system. The jobs are assumed to be malleable, that is, they can be executed with a variable number of processors and their parallelism also varies with time. At any time when the parallelism of a job is h, and the job is allocated a processors, the execution rate of the job is given by $\min\{a, h\}$. For each job J_i, let w_i denote the total work of the job, and let l_i denote the total length, or span of the job. The online algorithm is non-clairvoyant, so it must make scheduling decisions without knowing w_i, l_i and the parallelism profile of the job.

The problem is to design a scheduling algorithm that allocates a number of P processors through all the intermediate levels down to the jobs, and the objective is to minimize the overall completion time of the jobs, or the makespan. Note that when $K = 2$, the problem is reduced to the two-level problem that has been studied previously in [10]–[12]. Hence, our model represents a more general setting for the multiprocessor scheduling problem. Moreover, the tree structure is assumed to be given that reflects the characteristics of the system hierarchy. Hence, the scheduling algorithm cannot dynamically change the given structure to make better scheduling decisions. In addition, the scheduling decisions at the each individual node must also be made without knowing the corresponding decisions at other nodes. Therefore, the processors are allocated down the system hierarchy in a distributed and online manner.

We evaluate the performance of the online algorithm using both theoretical analysis and simulations. Theoretically, the performance is measured using competitive analysis [4], which compares the online algorithm with the optimal offline scheduler. Suppose for a job set $\mathcal{J}$, the makespan of the online algorithm is M, and the makespan of the optimal offline algorithm is M^*. Then the online algorithm is said to be c-competitive if $M \leq c \cdot M^* + b$, where b is a constant. For simulations, we extend Malleable-Lab [13], a tool we developed previously by augmenting Downeys parallel job model [16] with a set of generic parallelism variations, to evaluate the performance and the scalability of our proposed scheduling algorithms. Note that, to simply the theoretical analysis, we assume that all jobs are released in one batch with release time 0.

III. Two Adaptive Schedulers Based on Feedback-driven Policies

In this section, we present a framework for the hierarchical scheduling problem and derive performance bounds for two specific schedulers that rely on adaptive feedback-driven policies. In this framework, any scheduler can be used to indicate processor requirements for the jobs at the bottom level. In this paper, we use AG and ABG schedulers proposed by Agrawal et al. [11] and Sun et al. [10], respectively. At higher levels, we present a scheme, called Desire-Sum or DS for short, to aggregate the resources requirements at each intermediate level. Finally, we use the dynamic equi-partition (DEQ) algorithm [14] for allocating the processors resources.

A. Feedback-driven schedulers for jobs

AG and ABG schedulers calculate the processor requirement or the processor desire for each job periodically after a scheduling quantum expires. Both schedulers achieve this based on the execution status of the job in the previous quantum. Specifically, AG uses the processor utilization of the job and ABG uses the measured average parallelism of the job for the processor desire calculation.

1) AG: AG was first proposed by Agrawal et al [11]. Suppose in quantum q, job J_i requires for $d_i(q)$ processors and gets allocation of $a_i(q)$ processors. Then J_i is said to be satisfied if $a_i(q) \geq d_i(q)$, otherwise it is deprived. Moreover, suppose J_i completes $w_i(q)$ work in quantum q. Then J_i is said to be efficient if $w_i(q) \geq \delta a_i(q) L$, where L is the quantum length and $\delta < 1$ is the utilization threshold. Otherwise, J_i is

inefficient. The processor desire for job J_i in the next quantum is then calculated as shown below:

$$d_i(q+1) = \begin{cases} d_i(q) \cdot \rho & \text{if efficient and satisfied,} \\ d_i(q)/\rho & \text{if inefficient,} \\ d_i(q) & \text{if efficient and deprived,} \end{cases}$$

where $\rho > 1$ is the responsiveness parameter.

2) ABG: ABG was proposed by Sun et al. [10] after observing that the processor desires of AG tend to become unstable due to its multiplicative increase and multiplicative decrease nature. ABG on the other hand directly utilizes the average parallelism of the job in a quantum for the desire calculation, hence is much more stable and representative of the job's resource requirements. Suppose that job J_i completes $w_i(q)$ work and reduces its span by $l_i(q)$ in quantum q, then the average parallelism of the job is $A_i(q) = w_i(q)/l_i(q)$. The processor desire of J_i in the next quantum is set to the average parallelism $A_i(q)$:

$$d_i(q+1) = A_i(q).$$

The initial desire for both ABG and AG in the first scheduling quantum is set to be 1.

B. Desire aggregation scheme for nodes

We present a scheme that aggregates the processor desires from lower levels at each node of the intermediate level, and calculates an overall desire for reporting to the higher level. The scheme, called Desire-Sum or DS, collects the desires of the children nodes, process them by taking their aggregates as its own desire and reports that to the parent node.

Suppose the bottom level is denoted as level 1, and each intermediate node n_i^k at level $k > 1$ has c_i^k immediate children. At the end of each quantum q, the c_i^k children report their processor desires for quantum $q+1$ to node n_i^k, which are denoted as $\{d_{i1}(q+1), d_{i2}(q+1), \cdots\}$. Then node n_i^k calculates the aggregate desire $d_i(q+1)$ for quantum $q+1$ as shown below:

$$d_i(q+1) = \sum_{j=1}^{c_i^k} d_{ij}(q+1).$$

However, for the hierarchical scheduling problem, each level may not have the same quantum length. In this paper, we only consider the case where the quantum length at a particular level can only be an integral multiple of that at the immediate lower level. Suppose that the quantum at level k has not expired when nodes at level $k-1$ report their desires, then the aggregate desire calculated above will be discarded, and only the most recent desires from lower levels are taken when the quantum at level k does expire. This scheme is reasonable because when the number of levels increases, it is not likely that past execution characteristics of the jobs are still relevant to direct the future processor allocation.

C. Processor allocation algorithm

We apply the dynamic equi-partition (DEQ) algorithm [14] to allocate processors to the nodes at each level, including the jobs at the bottom level, based on their processor desires. Generally speaking, DEQ attempts to allocate an equal share of processors to each requesting node, but will not allocate more processors to a node than it desires.

Suppose that at the end of quantum q when the quantum for level k expires, and node n_i^k gets $a_i(q+1)$ processors at the beginning of quantum $q+1$. Then for the c_i^k children nodes under n_i^k, the equal processor share is $a_i(q+1)/c_i^k$. The processors are allocated to the children for quantum $q+1$ in the following way:

(1). For those children whose desires are not more than the equal share, their desires will be satisfied.

(2). Update the equal share by excluding the jobs already satisfied and the processors already allocated. Then satisfy those jobs whose desires now becomes less than or equal to the equal share. This process is repeated until no job can be easily satisfied.

(3). If all remaining jobs have more processor desire than the equal share, then the remaining processors will be shared equally among these jobs.

Note that this algorithm applies to all levels, including the jobs at the bottom level. At a particular level, it is executed only when the quantum for this level expires. However, lower levels could have smaller quantum length. Hence, the processors could be reallocated among the lower level nodes more frequently than the nodes at the higher level.

D. Performance analysis

Combining the schemes presented so far, we can get two different adaptive schedulers for the hierarchical scheduling problem, and we call them AG-DS and ABG-DS respectively. We provide performance bounds for the two schedulers in terms of the overall completion time or the makespan of the jobs in this subsection. The analysis uses the existing bounds for the AG and ABG schedulers, and they apply to the case where all levels share the same quantum length. Specifically, we show that the competitive ratios for the makespan of the jobs in this case are identical to the respective ratios obtained when there are only two levels in the hierarchy [10], [12]. In the next section, experimental studies are performed for the more general cases with different quantum length.

We first present the bound for AG based scheduler. The same analysis can be easily extended to the scheduler based on ABG at the bottom level. We first define two related concepts for the jobs. For any job J_i, let t_i denote the total satisfied time of the job, and let a_i denote the total deprived processor allocation for the job. Then the analysis from [11] gives the following bounds for t_i and a_i scheduled by AG:

$$t_i \leq \frac{2}{1-\delta} \cdot l_i + o(1), \quad (1)$$

$$a_i \leq \frac{1+\rho}{\delta} \cdot w_i, \quad (2)$$

where w_i and l_i are the work and the span of the job, respectively. Relying on these two bounds, we show the makespan ratio of AG-DS in the following theorem. Note that we assume that all jobs are released in a batch.

Theorem 1: For the hierarchical scheduling problem with the same quantum length in all levels, the makespan M of the jobs scheduled by AG-DS satisfies

$$M \leq \left(\frac{2}{1-\delta} + \frac{1+\rho}{\delta}\right) \cdot M^* + o(1), \tag{3}$$

where M^* denotes the optimal makespan of the jobs.

Proof: The performance is obtained by bounding the total satisfied time and the total deprived time of the last completed job separately.

Let J_k be the last completed job. Then the makespan of all jobs is the same as the completion time of J_k. The total satisfied time of J_k, according to Inequality (1), is given by $t_k \leq \frac{2}{1-\delta} \cdot l_k + o(1)$. When J_k is deprived, since all levels share the same quantum length, according to the desire aggregation scheme DS and the DEQ allocation policy, all the ancestors of job J_k, including the root node, in the hierarchy are also deprived. This is because if any ancestor of J_k is satisfied, then it could satisfy all its descendants, including J_k. Note that this property holds regardless of the number of hierarchical levels. Hence, all P processors must be allocated to the jobs in this case. Also based on Inequality (2), the total deprived time of job J_k is not more than $\frac{\sum a_i}{P} \leq \frac{1+\rho}{\delta} \cdot \frac{\sum w_i}{P}$.

The makespan of the jobs, which is the completion time of J_k, is then given by $M \leq \frac{2}{1-\delta} \cdot l_k + \frac{1+\rho}{\delta} \cdot \frac{\sum w_i}{P} + o(1)$. Because the optimal algorithm takes at least span l_k time to complete the job set, and $\frac{\sum w_i}{P}$ is also a lower bound on the makespan [12], the theorem is proved. ■

Theorem 1 shows that AG-based hierarchical scheduler is $O(1)$-competitive in terms of the makespan of the jobs, since ρ and δ can be considered as constants. We can now apply similar analysis to the ABG-based scheduler at the bottom level and achieve similar result. Specifically, the performance of ABG depends on a parameter, which is called the transition factor and denoted by $C \geq 1$ to indicate the maximum ratio on the average parallelism of the jobs over any two adjacent quanta. Let $A_i(q)$ and $A_i(q+1)$ denote the average parallelism of job J_i in quantum q and $q+1$, respectively. Then we have $\frac{1}{C} \leq \frac{A_i(q)}{A_i(q+1)} \leq C$. More details on this factor can be found in [10], [15]. The performance of ABG-DS is given in the following theorem.

Theorem 2: For the hierarchical scheduling problem with the same quantum length in all levels, the makespan M of the jobs scheduled by ABG-DS satisfies

$$M \leq 2(C+1) \cdot M^*, \tag{4}$$

where M^* denotes the optimal makespan of the jobs.

Proof: For ABG-DS, a job J_i is called over-allocated in a quantum q if the processor allocation is at least the average parallelism in the quantum, i.e., $a_i(q) \geq A_i(q)$. Otherwise it is called under-allocated. From the analysis in [10], the total over-allocated time t_i and the total under-allocated processor allocation a_i are given by $t_i \leq (C+1) \cdot l_i$ and $a_i \leq (C+1) \cdot w_i$, respectively. According to the ABG scheduler, job J_i is also deprived in a quantum if it is under-allocated. The rest of the analysis then follows exactly that of Theorem 1, and the bound can be easily derived. ■

Given any set of jobs, Theorem 2 shows that ABG-DS hierarchical scheduling algorithm achieves $O(1)$-competitive in terms of the makespan of the jobs given that the transition factor of the jobs are bounded by a constant. Moreover, Theorems 1 and 2 also show that both AG-DS and ABG-DS have the same competitive ratios as the corresponding two-level algorithms [10], [12]. This suggest that the two hierarchial scheduling algorithms achieve scalable performances in terms of makespan regardless of the number of hierarchical levels.

Finally, comparing ABG-DS and AG-DS, we can see that ABG-DS scheduler tends to have better performance when the jobs have smaller transition factor, hence smoother parallelism variations. In the next section, we study their performance experimentally for the multi-level setting as well as under variable quantum length over different levels.

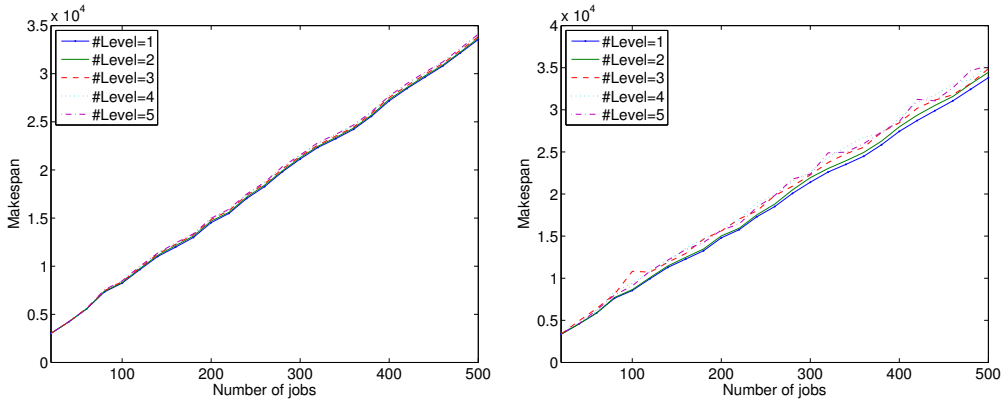
IV. Simulations

In this section, we conduct simulations to evaluate the performance of the hierarchical feedback-driven scheduling algorithms and compare them with EQUI-based scheduling policies. Specifically, we study the scalability, the performance with respect to makespan ratio, and the impact of different quantum patterns for these hierarchical scheduling algorithms.

A. Simulation setup

We extend an existing tool called Malleable-Lab [13] to support the hierarchical scheduling on multiprocessor systems. Malleable-Lab provides a flexible framework to evaluate the performance of online adaptive schedulers by integrating a detailed malleable job model augmented with a set of generic internal parallelism variations, which is essential for analyzing the dynamic behaviors of these adaptive schedulers. Based on Malleable-Lab, we build a multi-level scheduling framework, which is capable to divide the processor resource allocation into different levels. In addition, we implement a request-allotment protocol to support the feedback mechanism and processor resource allocation among different levels. Each level has its independent execution interval, or scheduling quantum, to aggregate performance requirements and adjust its child resource allotments.

In our study, we build a hierarchical system with 256 processors and the number of levels can be reset when starting to run the simulations. In this simulation framework, we implement three hierarchical scheduling policies: ABG-DS, AG-DS, and EQUI-EQUI, where EQUI-EQUI is the variation of well-known EQUI algorithm for hierarchical environment. In the following sections, we conduct several series of experiments respectively to study the scalability, performance, and impacts of scheduling quantum for these algorithms. To compare the performances of these different hierarchical schedulers, we use the following metrics: makespan, which is the completion time



(a) ABG-DS (b) AG-DS

Fig. 1. Scalability of ABG-DS and AG-DS with respect to makespan when increasing the number of levels.

of the last job in system, and makespan ratio, which is the ratio of online algorithm's makespan over that of the optimal offline algorithm given by the lower bound in the proof of Theorem 1. Following the traditional Downey's model, the system load is proportional with the number of jobs which increases from 20 to 500 with an increment of 20 each time. For any experiment we run it for 10 times with the same load and take the average result as the final output.

B. Simulation Results

(1) *scalability of feedback-driven scheduling policies*

Our first set of simulations focuses on studying the scalability of hierarchical feedback-driven schedulers. As shown in previous theoretical analysis, here the scalability of a hierarchical scheduling algorithm is a measure of its capacity to effectively respond to the changes of hierarchical levels. In our simulation, we evaluate the scalability of a given algorithm by adjusting the number of levels from 1 to 5 and the quantum length of each level is set to be the same. For each experiment, the simulation start under light load changing to heavy load which is identified by different job numbers. The simulation results of ABG-DS and AG-DS are shown in Fig. 1.

From the simulation results we can learn that both ABG-DS and AG-DS achieve better scalability with respect to changes in the number of levels. Moreover, as shown in Fig. 1, ABG-DS generally has more stable scalability than AG-DS in all cases. For example, when the number of levels is set to be 1 and 2, both ABG-DS and AG-DS have similar scalability in this case, but AG-DS exhibits slightly degrading scalability with increasing number of levels, as shown in Fig. 1b. The reason is that the task scheduler ABG provides a more stable and efficient feedback scheme than AG to calculate the processor requests, which finally leads to influencing the overall resource feedbacks. The simulation results indicate that the scheduling policies with stable feedback tend to achieve better scalability.

(2) *comparison with different scheduling policies*

In this section, we evaluate the performance of ABG-DS and AG-DS using the metric of makespan ratio and compare them with EQUI-EQUI, which equally divides all available processor resource among active jobs and internal nodes without any feedback information. We conduct a series of experiments by varying the number of levels from 1 to 5 and the quantum length of each level is set to be the same.

From the simulation results, as shown in Fig. 2, we can learn that the feedback-driven scheduling algorithms ABG-DS and AG-DS generally achieve better performance than EQUI-EQUI with respect to makespan ratio in all cases. Only when the system has light system load, ABG-DS and AG-DS are slightly worse than EQUI-EQUI. The reason is that feedback-driven scheduling strategies take advantage of the parallelism feedback based on the information of execution history while EQUI-EQUI is oblivious to job's parallelism and thus waste many processor resources. When the system has a small number of jobs, as shown in Fig. 2, EQUI-EQUI shows its advantages because in this case all the nodes and jobs can be easily satisfied with their processor requirements. Furthermore, as shown in Fig. 2, the performances of ABG-DS and AG-DS gradually tend to converge with that of EQUI-EQUI with the increasing system load because in this case any job can only receive very few processors most of time, and thus frequent processor reallocations have no obvious benefits.

The simulation results also demonstrate that ABG-DS has better performance than AG-DS for all system loads, especially when the number of levels increases. For example, when the number of levels is set to be 2, the makespan ratio of ABG-DS outperforms that of AG-DS by approximately 4% on average. When the number of levels is set to be 3, ABG-DS outperforms AG-DS by approximately 16% on average. Moreover, as shown in Fig. 2, the performance of ABG-DS is also more stable than AG-DS when changing the number of levels. Overall, feedback-driven scheduling algorithms have more advantages for scheduling malleable jobs whose internal parallelism is changing with time, and the better overall performance will be achieved with more stable

feedback mechanism.

(3) *Impact of scheduling quantum*

For adaptive scheduling algorithms, the scheduling quantum is an important system parameter, which may significantly affect the overall system performance. In this section, we consider the impact of scheduling quantum on these hierarchical scheduling algorithms. We conduct three sets of experiments following different scheduling quantum patterns. In the first set of experiments, the quantum length of all levels is set to be the same. In the second set, the quantum length of adjacent levels is increased by a factor of 2 from lower level to higher level. For instance, if the number of levels is set to be 5, the quantum length of each level from bottom to top is $L, 2L, 2^2L, 2^3L, 2^4L$ respectively, where L is the quantum length of lowest level. The quantum pattern of last set is similar to the second one but the factor is set to be 3. We evaluate these quantum patterns with different numbers of hierarchical levels and the simulation results are shown in Fig. 3. Note that the number of levels is only varied from 2 to 5, since when the number of level is 1 the quantum length does not changed with different quantum factor.

The simulation results show that the different quantum patterns has impacted on the performance of ABG-DS and AG-DS, but it is not significant for all cases, especially when the number of levels is not so large. From Fig. 3a and Fig. 3b we can learn that the performance of both ABG-DS and AG-DS is nearly unaffected by different quantum patterns when the number of levels is set to be 2. Specifically, the difference of makespan ratio for ABG-DS is only 0.5% between QuantumFactor=1 and QuantumFactor=2 on average and it is 1.1% averagely between QuantumFactor=1 and QuantumFactor=3. With the increasing number of levels, the impact of different quantum patterns is becoming slightly apparent. For example, the average difference of makespan ratio for ABG-DS reaches 11% between QuantumFactor=1 and QuantumFactor=2 when the number of level is 5. Moreover, the simulation results also demonstrate that AG-DS is more easily influenced than ABG-DS by different quantum patterns. Overall, as shown in simulation, the quantum patterns have impact on the system performance and the quantum length of higher level can not be set much larger than that of lower levels to guarantee good performance, especially when the number of level is high.

V. Conclusions

We have focused on the problem of online hierarchical scheduling to minimize the makespan for malleable parallel jobs. Non-clairvoyant scheduling is studied with emphasis on exploiting increasing malleability in parallelism for multicore- and multiprocessor-based systems. Integrating feedback-driven adaptive schemes, which are capable of dynamically adjusting the resource allotment, we propose two hierarchical scheduling algorithms: ABG-DS and AG-DS and show that both adaptive scheduling algorithms achieve scalable performance in term of makespan regardless of the number of hierarchical levels. The simulation results also demonstrate that feedback-driven hierarchical scheduling algorithms outperform EQUI-based policies for a wide range of workloads.

One of the future work is to extend the hierarchical adaptive scheduling model to more types of feedback-driven resource allotment policies and evaluate the performance of these online scheduling algorithms using both theoretical analysis and various parallel workloads. Another interesting work is to consider the different cost for interactions between the hierarchical levels since the delays can be successively longer when climbing up the tree.

Acknowledgment

This work is partially supported by China National Hi-tech Research and Development Program (863 Project) under the grants No. 2009DFA12110, 2009AA01Z108, 2009AA01A131 and Natural Science Foundation of China under the grant No.60873053. The authors would like to thank Wen-Jing Hsu for the initial idea of this work and helpful comments. Yangjie Cao also thanks Wen-Jing Hsu for generous hospitality when he was a visiting scholar at Nanyang Technological University during Summer 2009.

References

[1] P. Goyal, X. Guo, and H. Vin. A Hierarchical CPU Scheduler for Multimedia Operating Systems Proc. Second Usenix Symp. Operating System Design and Implementation (OSDI'96), pages 107–122, Oct. 1996.

[2] A. Chandra and P. Shenoy. Hierarchical scheduling for symmetric multiprocessors In *IEEE Transactions on Parallel and Distributed Systems*, pages 418–431, 2008.

[3] J.H. Abawajy. Adaptive hierarchical scheduling policy for enterprise grid computing systems *Journal of Network and Computer Applications*, 32(3):770–779, 2009.

[4] A. Borodin and R. El-Yaniv. *Online computation and competitive analysis*. Cambridge University Press, New York, NY, USA, 1998.

[5] R. Motwani, S. Phillips, and E. Torng. Non-clairvoyant scheduling. In *SODA*, pages 422–431, Austin, TX, USA, 1993.

[6] J. Edmonds. Scheduling in the dark. In *STOC*, pages 179–188, Atlanta, GA, USA, 1999.

[7] Jeff Edmonds and Donald D. Chinn and Tim Brecht and Xiaotie Deng. Non-clairvoyant Multiprocessor Scheduling of Jobs with Changing Execution Characteristics. In *Journal of Scheduling*, 6(3):231–250, 2003.

[8] J. Robert and N. Schabanel. Non-clairvoyant batch set scheduling: Fairness is fair enough. In *ESA*, pages 741–753, Eilat, Israel, 2007.

[9] H. Sun, Y. Cao, and W.-J. Hsu. Non-clairvoyant Adaptive Scheduling for Sets of Parallel Jobs with Fairness and Efficiency. Submitted.

[10] H. Sun, and W.-J. Hsu. Adaptive B-Greedy (ABG): A Simple yet Efficient Scheduling Algorithm. In *SMTPS* in conjunction with *IPDPS*, pages 1–8, Miami, FL, USA, 2008.

[11] K. Agrawal, Y. He, W.-J. Hsu, and C. E. Leiserson, Adaptive scheduling with parallelism feedback, In *PPoPP*, pages 100 – 109, New York City, NY, USA, 2006.

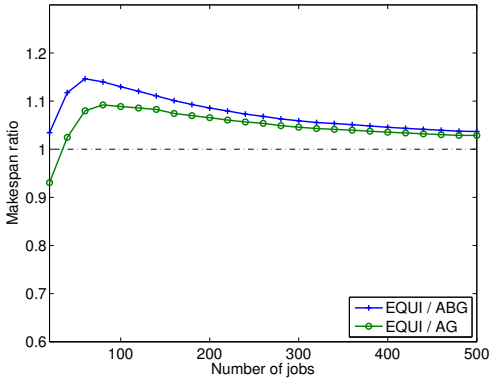
[12] Y. He, W.-J. Hsu, and C. E. Leiserson. Provably efficient two-level adaptive scheduling. In *JSSPP*, pages 1–32, Saint-Malo, France, 2006.

[13] Y. Cao, H. Sun, and W.-J. Hsu. Malleable-Lab: A tool for evaluating adaptive online schedulers on malleable jobs. *The 18th Euromicro International Conference on Parallel, Distributed and Network-Based Computing*, pages 11-18, Piza, Italy, 2010.

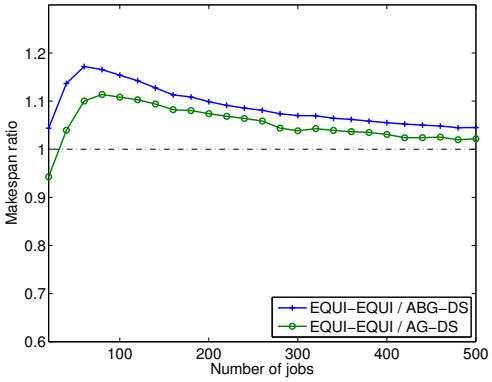
[14] C. McCann, R. Vaswani, and J. Zahorjan. A dynamic processor allocation policy for multiprogrammed shared-memory multiprocessors. *ACM Transactions on Computer Systems*, 11(2):146–178, 1993.

[15] H. Sun, Y. Cao, and W.-J. Hsu. Efficient Adaptive Scheduling of Multiprocessors with Stable Parallelism Feedback. To appear in *IEEE Transactions on Parallel and Distributed Systems*.

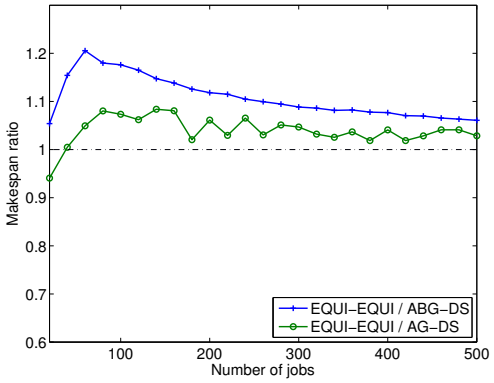
[16] A. B. Downey. A parallel workload model and its implications for processor allocation. In *HPDC*, page 112, Portland, OR, USA, 1997.



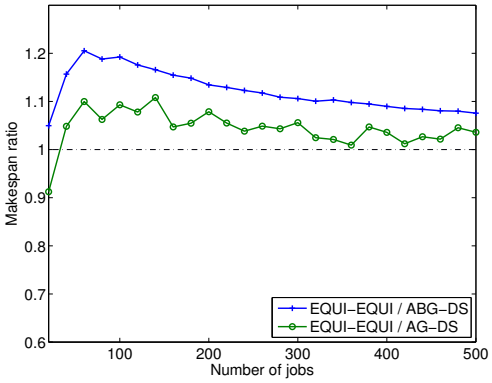
(a) number of levels = 1



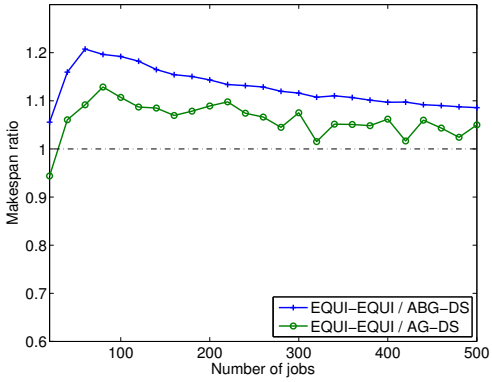
(b) number of levels = 2



(c) number of levels = 3

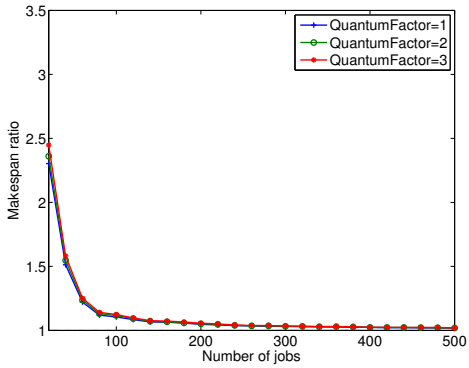


(d) number of levels = 4

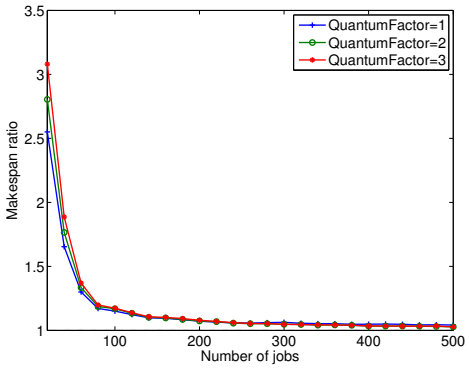


(e) number of levels = 5

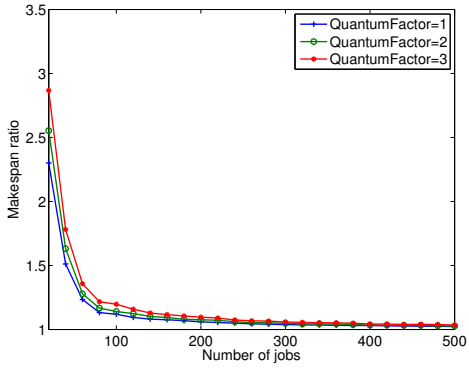
Fig. 2. Makespan comparisons of ABG-DS and AG-DS with EQUI-EQUI.



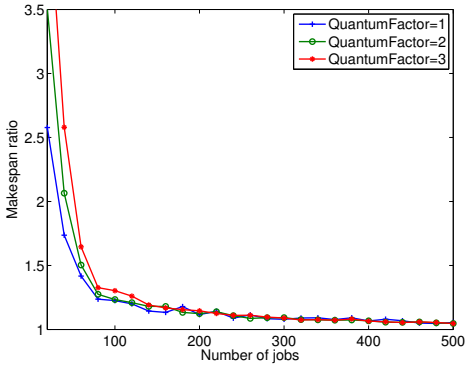
(a) ABG-DS (number of levels = 2)



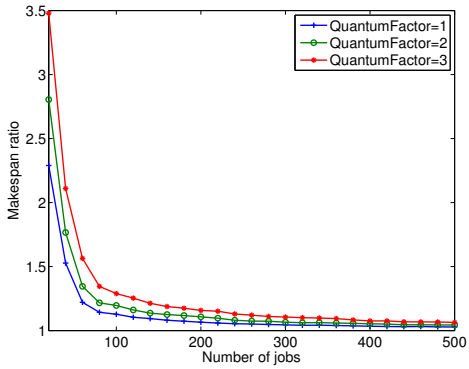
(b) AG-DS (number of levels = 2)



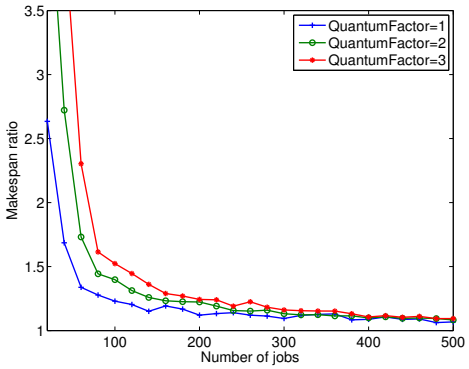
(c) ABG-DS (number of levels = 3)



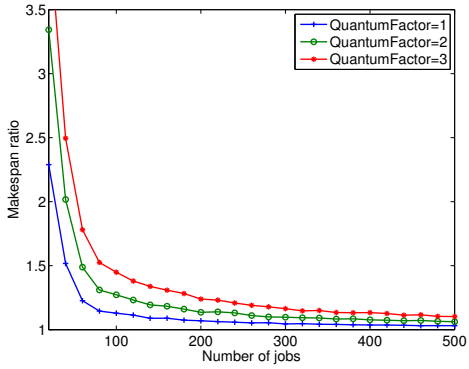
(d) AG-DS (number of levels = 3)



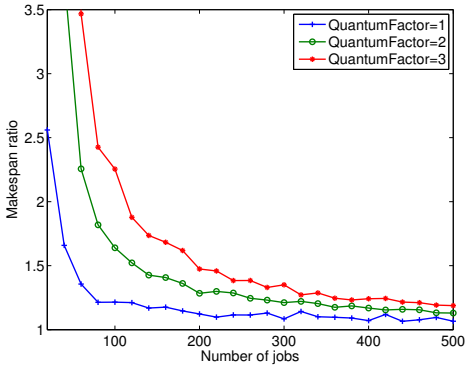
(e) ABG-DS (number of levels = 4)



(f) AG-DS (number of levels = 4)



(g) ABG-DS (number of levels = 5)



(h) AG-DS (number of levels = 5)

Fig. 3. Impact of different quantum patterns on ABG-DS and AG-DS with respect to makespan ratio.