Improved Parallel Job Scheduling with Overhead

Jessen T. Havill
Department of Mathematics and Computer Science, Denison University, Granville, OH 43023, USA
havill@denison.edu

Weizhen Mao
Department of Computer Science, College of William and Mary, Williamsburg, VA 23187–8795, USA
wm@cs.wm.edu

Vesselin Dimitrov
Department of Mathematics and Computer Science, Denison University, Granville, OH 43023, USA
dimitr_v@denison.edu

Abstract

We consider a parallel job scheduling model that incorporates both computation time and communication overhead. For any job J_j with length p_j, if k_j processors are assigned to execute the job, then the actual execution time of the job is $t_j = p_j/k_j + (k_j - 1)c$, where c is a constant overhead cost associated with each processor except the master processor that initiates the parallel computation. Previously, it was shown that the Shortest Execution Time (SET) algorithm has competitive ratio $4(m-1)/m$ for even $m \geq 2$ and $4m/(m+1)$ for odd $m \geq 3$ with respect to makespan. Here we study the Earliest Completion Time (ECT) algorithm, and show that its competitive ratio is 2 and 2.25 on 2 and 3 processors, respectively. We also offer simulation results that show that ECT compares favorably to SET on larger numbers of processors. Finally, we show that any online algorithm for our problem has competitive ratio at least 3/2 for arbitrarily large m.

Keywords: Parallel job scheduling, online algorithms, competitive ratio, simulation.

1 Introduction

In a parallel system, multiple processors are available to collectively execute incoming jobs. We assume that jobs are *malleable*, i.e., they can be executed by any number of processors. Ideally, the computation time needed for a malleable job should be inversely proportional to the number of processors assigned to it. However, an overhead cost is also incurred due to the communication and coordination needs of the processors. This contributes to a slowdown that is often proportional to the number of processors assigned. We consider the following simple mathematical model which incorporates both the computation speedup and the overhead slowdown. In this model, we define a constant c to be the overhead cost associated with each processor that is used to parallelize the computation. Then if a job J_j with length (processing requirement) p_j is assigned to k_j processors, its execution time is $t_j = p_j/k_j + (k_j - 1)c$. Notice that $t_j = p_j$ when $k_j = 1$. The work w_j of a job J_j is defined to be the product of the number of processors assigned to the job and the execution time of the job. Therefore, $w_j = k_j t_j = p_j + k_j(k_j - 1)c \geq p_j$.

We study the following optimization problem defined under this model. In a parallel system with m identical processors, a sequence of n independent jobs $J_1, \ldots, J_n$ are to be executed, where job J_j has a length of p_j. To schedule job J_j, an algorithm must determine k_j, the number of processors assigned to the job, and s_j, the start time of the job on the k_j selected processors. Once k_j and s_j are fixed, the execution time of the job is $t_j = p_j/k_j + (k_j - 1)c$ and the completion time of the job is $C_j = s_j + t_j$. The goal of the problem is to construct a schedule of the jobs that minimizes the makespan (maximum completion time) $\max_j C_j$. In order to make an algorithm useful in practice, we require scheduling decisions to be made online: the jobs are scheduled in order, and when scheduling J_j, nothing is known about $J_{j+1}, \ldots, J_n$. The competitive ratio of an online algorithm is defined to be the maximum ratio, among all instances, of the makespan of the schedule constructed by the online algorithm, denoted C, to the makespan of the optimal schedule (OPT), denoted C^*.

Job scheduling has been a fruitful area of research for many decades [5]. The study of parallel job scheduling has recently drawn a lot of attention from researchers with the rapid development of parallel systems (e.g., see [1, 2, 9]). Our formulation (with $t_j = p_j/k_j + (k_j - 1)c$) is a variation of the models offered by Sevcik in [8]. It accurately models jobs exhibiting data parallelism in shared memory architectures [6]. Data parallelism techniques are very common and can be applied to general vector and polynomial computations, sorting and searching, and matrix multiplication.

2 The ECT algorithm

The online algorithm we study in this paper is called the Earliest Completion Time (ECT) algorithm. To schedule J_j, ECT chooses a value of k_j that minimizes the completion time of J_j.

Since this parallel job scheduling model was proposed in [7], an online algorithm called the Shortest Execution Time (SET) algorithm has received some attention [3, 4]. To schedule job J_j, SET chooses k_j so that it minimizes the execution time t_j, and then schedules the job as early as possible. This value of k_j is either the floor or ceiling of $\sqrt{p_j/c}$ if $\sqrt{p_j/c} < m$, or m if $\sqrt{p_j/c} \geq m$. Havill and Mao [4] proved that the competitive ratio of SET is $4(m-1)/m$ for even $m \geq 2$ and $4m/(m+1)$ for odd $m \geq 3$.

In the next two sections, we compare the performance of ECT with SET in two ways: (1) by competitive ratio for small values of m, and (2) by simulation for large values of m. We shall see strong indications that ECT outperforms SET by a reasonable margin both analytically and in practice. In Section 5, we show that any online algorithm for our problem has competitive ratio at least 3/2 for arbitrarily large m.

3 Competitive ratios for $m = 2, 3$

Next we show that the competitive ratio of ECT is 2 when $m = 2$ and 9/4 when $m = 3$. We note that the competitive ratio of SET is also 2 when $m = 2$, but the competitive ratio of SET is 3 when $m = 3$ (see [4]).

Theorem 1. *The competitive ratio of ECT is* 2 *when* $m = 2$.

Proof First we prove the lower bound by showing that there is an instance for which $C/C^* = 2$. Consider an instance of $n = 2\alpha$ jobs, for integer $\alpha > 0$, with $p_j = 2c + \varepsilon$, where $\varepsilon > 0$ is arbitrarily small. In the ECT schedule, all jobs are assigned to two processors, yielding makespan

$$C = n((2c+\varepsilon)/2 + c) = n(2c+\varepsilon/2).$$

In the OPT schedule, all jobs are assigned to one processor, without any idle time, yielding makespan

$$C^* = (n/2)(2c+\varepsilon).$$

Thus the competitive ratio of ECT is at least

$$C/C^* = 2(2c+\varepsilon/2)/(2c+\varepsilon),$$

which approaches to 2 as $\varepsilon \to 0$.

Next we prove the upper bound by showing that, for any instance, $C/C^* \leq 2$. Recall that in any schedule, if job J_j is assigned to k_j processors, then the work performed by the job is defined to be $w_j = k_j t_j = p_j + k_j(k_j-1)c \geq p_j$. Let W and W^* be the total work, for all jobs, in the ECT schedule and the OPT schedule (for the same instance), respectively. Let I and I^* be the total idle time, on all processors, in the ECT and OPT schedules, respectively. Then, we have $C = (W+I)/m$ and $C^* = (W^*+I^*)/m \geq W^*/m \geq \sum_j p_j/m$. Therefore, $C/C^* \leq (W+I)/\sum_j p_j$. To prove that $C/C^* \leq 2$, we only need to show that $W + I \leq 2\sum_j p_j$.

For any instance, let S_1 and S_2 be the sets of jobs assigned to one and two processors by ECT, respectively. Since the two processors will never be idle simultaneously in an ECT schedule, we know that $I \leq \sum_{S_1} w_j = \sum_{S_1} p_j$. Also, $W = \sum_{S_1} w_j + \sum_{S_2} w_j = \sum_{S_1} p_j + \sum_{S_2}(p_j + 2c) \leq \sum_{S_1} p_j + 2\sum_{S_2} p_j$, since $p_j \geq 2c$ for $J_j \in S_2$ due to how ECT chooses k_j. Therefore, $W + I \leq 2\sum p_j$. □

Theorem 2. *The competitive ratio of ECT is* $9/4$ *when* $m = 3$.

Proof Sketch First we prove the lower bound. Consider an instance of $n = 6\alpha$ jobs, for any integer $\alpha > 0$, with $p_j = 2c + \varepsilon$ for odd j and $p_j = 6c + \varepsilon$ for even j, where $\varepsilon > 0$ is arbitrarily small. In the ECT schedule, all odd-indexed jobs are assigned to two processors and all even-indexed jobs are assigned to three processors, yielding makespan

$$C = (2c + \varepsilon/2 + 4c + \varepsilon/3)n/2 = (3c + 5\varepsilon/12)n.$$

In the OPT schedule, all jobs are assigned to one processor, without any idle time, yielding makespan

$$C^* = (2c + \varepsilon + 6c + \varepsilon)n/6 = (4c/3 + \varepsilon/3)n.$$

Thus the competitive ratio of ECT is at least

$$C/C^* = (3c + 5\varepsilon/12)/(4c/3 + \varepsilon/3),$$

which approaches $9/4$ as $\varepsilon \to 0$.

Due to space considerations, we will only sketch a proof of the upper bound here. We partition the ECT schedule into blocks, where each block B_i is defined to be a maximal time interval $[b_i, b_{i+1})$ during which no job starts executing. Let W_i be the amount of work performed in block B_i in the ECT schedule and let I_i be the total idle time, on all processors, in block B_i in the ECT schedule. Also, let W_i^* be the amount of work performed in the OPT schedule to execute the jobs (and/or partial jobs) present in block B_i in the ECT schedule. Let $W_{i,j}$, $I_{i,j}$, and $W_{i,j}^*$ each denote the sum of the respective quantity over consecutive blocks $B_i, \ldots, B_j$. We partition the ECT schedule into groups of consecutive blocks and show that

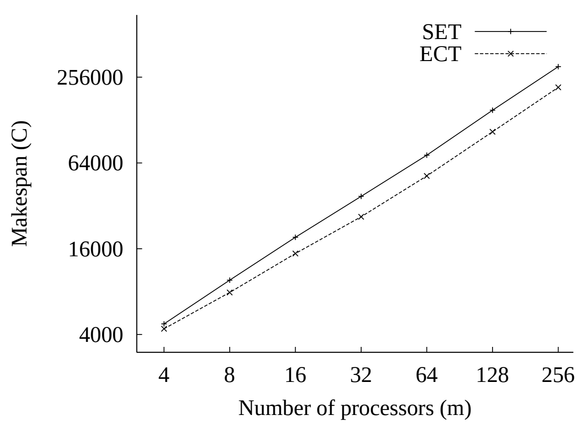

Figure 1: Comparison of the makespans of the ECT and SET schedules.

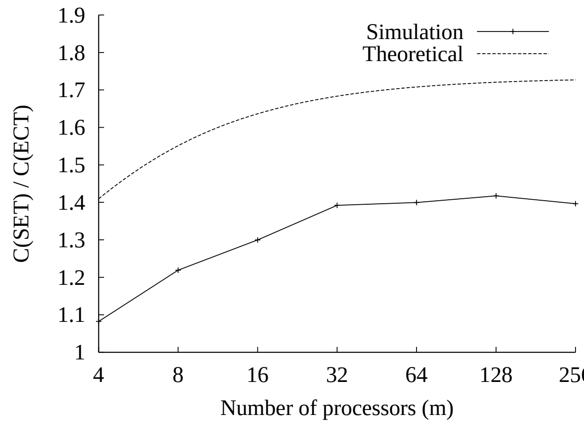

Figure 2: Ratio of the SET makespan to the ECT makespan.

$(W_{i,j}+I_{i,j})/W^*_{i,j} \leq 9/4$ for each group. From this the upper bound result follows:

$$\begin{aligned} \frac{C}{C^*} &= \frac{W+I}{W^*+I^*} \leq \frac{W+I}{W^*} = \frac{\sum(W_{i,j}+I_{i,j})}{\sum W^*_{i,j}} \\ &\leq \frac{\sum(9W^*_{i,j}/4)}{\sum W^*_{i,j}} = \frac{9}{4}\,. \end{aligned}$$

We prove the inequality $(W_{i,j}+I_{i,j})/W^*_{i,j} \leq 9/4$ for each of the following three cases. In the first case, we consider a block B_i in which two processors are idle at the end of the block. In the second case, we consider a block B_i in which only one processor is idle at the end of the block. In the third case, we consider a block B_i with no idle area. □

We note that the lower bound in Theorem 2 can be generalized to $(30m-36)/(13m-15)$ for any $m \geq 3$ that is divisible by 3 by making $p_j = (2m/3)(2m/3-1)c+\varepsilon$ for odd j and $p_j = m(m-1)c+\varepsilon$ for even j. We conjecture that a matching upper bound holds as well, which would make the competitive ratio of ECT equal to $30/13 \approx 2.308$ for arbitrary large m.

4 Simulations for $m \geq 4$

We next present simulation results that compare ECT and SET. The simulations were run on 1000 jobs with lengths that were uniformly randomly generated between 1 and $m(m+1)$, assuming $c = 1$. The simulation was run with exponentially increasing values of m between 4 and 256. Figure 1 compares the makespan of the schedules constructed by ECT and SET. Both axes in the plot use a logarithmic scale so the two curves are differentiable for small values of m. Figure 2 plots the ratio between the makespan of the SET schedule and the ECT schedule. This ratio ranges from 1.08 for $m = 4$ to about 1.4 for $m = 32$ to $m = 256$. The dashed curve in Figure 2 shows the ratio between the proven competitive ratio of SET and the conjectured competitive ratio of ECT. The reason that the simulation curve is below the theoretical curve is that the simulation curve reflects the average case while the theoretical curve reflects the worst case. The fact that the curves shadow each other provides good evidence that our conjecture is correct.

5 A general lower bound

Next we prove a general lower bound on the competitive ratio for any online algorithm for our problem. This general lower bound reveals how difficult it is for an online algorithm to solve the scheduling problem approximately.

Theorem 3. *The competitive ratio of any online algorithm for parallel job scheduling is least* $3/2$ *asymptotically.*

Proof We frame the proof as a contest between an online algorithm A and an online adversary which issues the job requests. The adversary begins by issuing a job with size pc. Suppose A chooses to assign this first job to k processors, for $1 \leq k \leq \lfloor m/2 \rfloor$. Then the adversary assigns the job to m processors and stops. In this case, the competitive ratio is at least

$$\min_{1\leq k\leq\lfloor m/2\rfloor} \left\{ \frac{pc/k+(k-1)c}{pc/m+(m-1)c} \right\}\,. \tag{1}$$

On the other hand, if A chooses to assign the first job to k processors, for $\lfloor m/2 \rfloor < k \leq m$, then the adversary assigns the job to one processor, issues $m-1$ more jobs of size pc, and puts each of these $m-1$ jobs on one processor as well in parallel with the first. In this second case, the competitive ratio is at least

$$\min_{\lfloor m/2\rfloor < k\leq m}\left\{\frac{m(pc/k+(k-1)c)}{pc}\right\}. \quad (2)$$

Therefore, the competitive ratio of A is at least the minimum of (1) and (2). Notice that $(p/k+(k-1))/(p/m+(m-1))$, for $1\leq k\leq \lfloor m/2\rfloor$, is increasing with respect to p and $m(p/k+(k-1))/p$, for $\lfloor m/2\rfloor < k\leq m$, is decreasing with respect to p. The minimum of (1) and (2) is minimized when the slowest increasing function $(p/\lfloor m/2\rfloor+(\lfloor m/2\rfloor-1))/(p/m+(m-1))$ intersects with the fastest decreasing function $m(p/m+(m-1))/p$. This occurs when p is

$$p^* = \frac{(2m-\lfloor m/2\rfloor-1)}{2(1/\lfloor m/2\rfloor-1/m)}+$$

$$\frac{\sqrt{(2m-\lfloor m/2\rfloor-1)^2+4m(m-1)^2(1/\lfloor m/2\rfloor-1/m)}}{2(1/\lfloor m/2\rfloor-1/m)}.$$

Substituting $p=p^*$ into the minimum of (1) or (2), we get a lower bound of

$$\begin{cases} \dfrac{7m-6+\sqrt{25m^2-44m+20}}{3m-2+\sqrt{25m^2-44m+20}}, & \text{if } m \text{ is even} \\[2ex] \dfrac{7m+3+\sqrt{25m^2-6m-15}}{3m-1+\sqrt{25m^2-6m-15}}, & \text{if } m \text{ is odd} \end{cases}.$$

Notice that, when $m=2$, this lower bound becomes $\sqrt{2}$. When m is arbitrarily large, this bound approaches 3/2 for both even and odd m. □

6 Conclusions

In this paper, we continued our work on a simple model for parallel job scheduling that includes the consideration of an overhead cost among processors working on a job simultaneously. We studied an online algorithm, ECT, that always assigns a number of processors to a job that will minimize its completion time. We proved the tight competitive ratios of 2 and $9/4$ for $m=2$ and $m=3$ processors, respectively. In the case when $m\geq 4$, we used simulation to show that, on average, ECT outperforms another online algorithm, SET, that assigns the number of processors to a job that will minimize the job's execution time. In addition, we showed that the competitive ratio of any online algorithm is at least 1.5 for arbitrarily large m.

In the future, we are interested in proving a tight competitive ratio for ECT when $m\geq 4$. From the results presented in this paper, we strongly suspect that the tight competitive ratio of ECT approaches 30/13 when m is arbitrarily large.

References

[1] J. Du and J. Y-H. Leung, Complexity of scheduling parallel task systems, *SIAM Journal on Discrete Mathematics* 2, 473–487 (1989).

[2] D. G. Feitelson, L. Rudolph, U. Schwiegelshohn, K. Sevcik, and P. Wong, Theory and practice in parallel job scheduling, *Job Scheduling Strategies for Parallel Processing*, D. G. Feitelson and L. Rudolph (eds.), Lecture Notes in Computer Science, Springer Verlag (1997).

[3] J. T. Havill, A competitive online algorithm for a parallel job scheduling problem, *Proceedings of the 12th IASTED International Conference on Parallel and Distributed Computing and Systems*, 611–616 (2000).

[4] J. T. Havill and W. Mao, Competitive online parallel job scheduling with overhead, *Manuscript*.

[5] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys, Sequencing and scheduling: Algorithms and complexity, *Handbooks in Operations Research and Management Science, Volume 4: Logistics of Production and Inventory*, edited by So. C. Graves, A. H. Rinnooy Kan, and P. Zipkin, North-Holland (1990).

[6] B. P. Lester, *The Art of Parallel Programming*, Prentice Hall, 1993.

[7] W. Mao, J. Chen, and W. Watson III, On-line algorithms for a parallel job scheduling problem, *Proceedings of the 11th IASTED International Conference on Parallel and Distributed Computing and Systems*, 753–757 (1999).

[8] K. C. Sevcik, Application scheduling and processor allocation in multiprogrammed parallel processing systems, *Performance Evaluation*, 19, 107–140 (1994).

[9] J. Sgall, On-line scheduling of parallel jobs, *Proceedings of the International Symposium on Mathematical Foundations of Computer Science*, LNCS 841, 159–176 (1994).