Federated Scheduling Admits No Constant Speedup Factors for Constrained-Deadline DAG Task Systems[★]

Jian-Jia Chen

Department of Informatics, TU Dortmund University, Germany

Abstract. In the federated scheduling approaches in multiprocessor systems, a task either 1) is restricted to execute sequentially on a single processor or 2) has exclusive access to the assigned processors. There have been several positive results to conduct good federated scheduling policies, which have constant speedup factors with respect to any optimal federated scheduling algorithm. This paper answers an open question: "For constrained-deadline task systems with directed acyclic graph (DAG) dependency structures, do federated scheduling policies have a constant speedup factor with respect to any optimal scheduling algorithm?" The answer is "No!" This paper presents an example, which demonstrates that any federated scheduling algorithm has a speedup factor of at least $\Omega(\min\{M, N\})$ with respect to any optimal scheduling algorithm, where N is the number of tasks and M is the number of processors.

1 Introduction

The sporadic task model has been widely adopted in real-time systems. In the sporadic task model, a task τ_i is characterized by its relative deadline D_i, its minimum inter-arrival time T_i. A sporadic task is an infinite sequence of task instances, referred to as *jobs*, where two consecutive jobs of a task should arrive no shorter than the minimum inter-arrival time separation. A sporadic task system τ is called an implicit-deadline system if $D_i = T_i$ holds for each τ_i. A sporadic task system τ is called a constrained-deadline system if $D_i \leq T_i$ holds for each τ_i. Otherwise, such a sporadic task system τ is an arbitrary-deadline system.

Traditionally, each task τ_i is also associated with its worst-case execution time (WCET) C_i. In uniprocessor platforms, in the literature, since the processor only executes one job at one time point, there is no need to express potential parallel execution paths. Multiprocessor platforms allow *inter-task parallelism*, which enables the capability to execute sequential programs concurrently, and *intra-task parallelism*, which allows a parallelized task to be executed in parallel at the same time. The parallelism of a job of a task can be represented by a *directed acyclic graph (DAG)*. That is, the DAG structure defines the precedence constraints of the subtasks. In other words, the execution of task τ_i can be divided into subtasks and the precedence constraints of these subtasks are defined by a DAG structure.

To handle a set of DAG tasks on multiprocessor platforms, the recent studies by Li et al. [6] and Baruah [1–3] suggest to use federated scheduling. In the federated scheduling

---

in multiprocessor systems, a task 1) either is restricted to execute sequentially on a single processor or 2) has exclusive access to the assigned processors. The federated scheduling strategy was originally proposed in [6] for implicit-deadline task systems. Baruah [1–3] adopted the concept of federated scheduling for constrained-deadline and arbitrary-deadline task systems.

A scheduling algorithm generates a schedule for the task system τ to define when and how the jobs are executed on the platform. A schedule is *feasible* if no job misses its deadline and all the scheduling constraints are respected (i.e., precedence constraints, minimum inter-arrival time between two consecutive jobs of a task, etc.). An optimal scheduling algorithm is defined as follows: *If there exist a feasible schedule, an optimal scheduling algorithm produces one of them.* Similarly, an optimal <u>federated</u> scheduling algorithm is defined as follows: *If there exist feasible <u>federated</u> schedules, an optimal <u>federated</u> scheduling algorithm produces one of them.*

The federated scheduling strategies are not optimal scheduling strategies. That is, there exist task sets which can be feasibly scheduled to meet their deadlines, but the federated scheduling strategies lead to deadline misses while scheduling those task sets. One widely-adopted theoretical measure to quantify the approximation made in such *non-optimal* scheduling strategies is the *speedup factors*, defined as follows:

Definition 1. *A scheduling algorithm $\mathcal{A}$ is said to have a speedup factor s with respect to a scheduling algorithm $\mathcal{B}$ if the following condition always holds:*

 – *For any task system τ that can be feasibly scheduled by the scheduling algorithm $\mathcal{B}$, the schedule derived from the scheduling algorithm $\mathcal{A}$ is feasible by speeding up (each of the processors) to s times as fast as in the original platform (speed).* □

The quantitive measure of speedup factors is always related to the reference scheduling algorithm $\mathcal{B}$. The speedup factor with respect to any optimal scheduling algorithm provides an *absolute* measure to evaluate the theoretical gap of the scheduling algorithm $\mathcal{A}$. However, if $\mathcal{B}$ is only a sub-optimal scheduling algorithm, the speedup factor with respect to $\mathcal{B}$ provides only a *relative* measure. Therefore, if the reference algorithm $\mathcal{B}$ is very far from any optimal scheduling algorithm, the quantitive speedup factors with respect to $\mathcal{B}$ may be misleading.

The existing results of speedup factors for federated scheduling on M identical processors can be summarized as follows:

 – The speedup factor of the federated scheduling algorithm in [6] for implicit-deadline task systems in identical multiprocessor platforms is 2 with respect to *any optimal scheduling algorithm.*[1]
 – The speedup factor of the federated scheduling algorithms in [1, 3] for constrained-deadline task systems in identical multiprocessor platforms is $3-1/M$ with respect to *any optimal federated scheduling algorithm.*
 – The speedup factor of the federated scheduling algorithms in [2, 3] for arbitrary-deadline task systems in identical multiprocessor platforms is $4-2/M$ with respect to *any optimal federated scheduling algorithm.*

---

[1] The paper [6] uses another quantification metric, called capacity augmentation factor. It is also shown that a capacity augmentation factor 2 also implies a speedup factor 2 for implicit-deadline task systems.

	τ_1	τ_2	τ_3	τ_4	τ_5	τ_6	τ_7	τ_9	τ_{10}
C_i	10	10	20	40	80	160	320	640	1280
D_i	1	2	4	8	16	32	64	128	256
T_i	∞								

Table 1: An example of the task set τ when $N = 10$, $M = 10$, and $K = 2$.

Therefore, there is a potential gap between the *relative* speedup factors used in [1–3] (with respect to any optimal federated scheduling algorithm) and the *absolute* speedup factors with respect to any optimal scheduling algorithm. The results in [1–3] can only be concluded to have a constant speedup factor with respect to any optimal scheduling algorithm if the federated schedules have a constant speedup factor with respect to any optimal scheduling algorithm. It could be possible that federated scheduling itself is not a good scheduling strategy (with respect to any optimal scheduling algorithm). If so, the constant federated speedup factors in [1–3] can be misleading and do not result in constant speedup factors with respect to any optimal scheduling algorithm.

For constrained-deadline task systems with DAG, the contribution of this paper in Section 2 shows that "the speedup factor of any federated scheduling algorithm with respect to any optimal scheduling algorithm is at least $\Omega(\min\{M, N\})$, where N is the number of tasks and M is the number of processors." This concludes that the speedup factors (with respect to any optimal scheduling algorithm) of the algorithms in [1–3] are at least $\Omega(\min\{M, N\})$. However, please note that the result in this paper does not invalidate the constant speedup factors with respect to any optimal federated scheduling, as claimed in [1–3].

2 Speedup Factor Lower Bound of Federated Scheduling

To prove the lower bound of the speedup factors of any federated scheduling algorithm with respect to any optimal scheduling algorithm, we just have to show that there exist input task sets that admit feasible schedules but cannot be feasibly scheduled by any federated scheduling strategies under a constant speedup factor. Specifically, in the provided input task set, it is not necessary to exploit any specific DAG constraints. The lower bound is built based on the observation of the pessimistic strategy in federated scheduling to *grant a task exclusive access to the processors upon which they execute if the task needs more than one processor*.

Suppose that $M \geq 2$ is a positive integer. Moreover, let K be any arbitrary number with $K \geq 2$. We create N sporadic tasks with the following setting:

- $C_1 = M$, $D_1 = 1$, and $T_1 = \infty$.
- $C_i = K^{i-2}(K - 1)M$, $D_i = K^{i-1}$, and $T_i = \infty$ for $i = 2, 3, \ldots, N$.

Table 1 provides a concrete example when N is 10, M is 10 and K is 2.

We assume that each task τ_i has M subtasks and there is no precedence constraint among these M subtasks (a special case of DAG). Each subtask of task τ_i has the worst-case execution time $\frac{C_i}{M}$. For the rest of this section, we denote this task set as $\tau_{counter}$.

3

We will first show in Lemma 1 that task set $\tau_{counter}$ admits feasible schedules.

Lemma 1. *There exists a feasible schedule of the given task set $\tau_{counter}$.*

Proof. Since each task τ_i has M (independent) subtasks with the same execution time, we can greedily assign each of them to one processor *statically* and apply the earliest-deadline-first (EDF) scheduling algorithm individually on each of the M processors. Therefore, the feasibility of the schedule can be easily verified by validating whether the subtasks on one processor can meet the deadline or not. This can be verified by using the demand bound function analysis provided by Baruah et al. [4]. Since $\frac{\sum_{i=1}^{j} C_i}{M} = D_j$ for $j = 1, 2, \ldots, N$, the above schedule is a feasible one. □

The following lemma shows that task set $\tau_{counter}$ cannot be feasibly scheduled by any federated scheduling algorithm if the speedup factor s is not big enough.

Lemma 2. *Suppose that the M processors are speeded up to s times of the original speed, where s is strictly smaller than $(1 - \frac{1}{K})M$, i.e., $s < (1 - \frac{1}{K})M$. A federated schedule for task set $\tau_{counter}$ requires at least $\frac{M}{s}\left(N - \frac{N-1}{K}\right)$ processors with speed s to feasibly schedule the given task set $\tau_{counter}$. That is, if $s < (1 - \frac{1}{K})M$ and $\frac{M}{s}\left(N - \frac{N-1}{K}\right) > M$, then there is no feasible federated schedule for task set $\tau_{counter}$ on M processors at such a speed s.*

Proof. If $\frac{C_i}{D_i s} > 1$, the concept of federated scheduling, i.e., *tasks that are permitted to execute upon more than one processor are granted exclusive access to the processors upon which they execute*, would need to execute task τ_i exclusively on at least $\left\lceil \frac{C_i}{D_i s} \right\rceil$ processors at speed s *exclusively* to serve task τ_i.

For task τ_1, at least $\left\lceil \frac{C_1}{D_1 s} \right\rceil > \frac{1}{1 - \frac{1}{K}} > 1$ processors are needed to ensure the feasibility of task τ_1. Moreover, for $i = 2, 3, \ldots, N$, we have $\left\lceil \frac{C_i}{D_i s} \right\rceil \geq \frac{K^{i-2}(K-1)M}{K^{i-1}s} = (1 - \frac{1}{K})\frac{M}{s} > 1$. Therefore, the assumption $s < (1 - \frac{1}{K})M$ implies that a federated scheduling algorithm has to run these N tasks exclusively on the granted processors. So, task τ_i is assigned to be executed on at least $\left\lceil \frac{C_i}{D_i s} \right\rceil$ dedicated processors. Therefore, if $s < (1 - \frac{1}{K})M$, the number of processors in federated scheduling requires at least

$$\sum_{i=1}^{N} \left\lceil \frac{C_i}{D_i s} \right\rceil \geq \frac{M}{s} + \sum_{i=2}^{N} \frac{K^{i-2}(K-1)M}{K^{i-1}s} = \frac{M}{s} + \frac{M}{s} \sum_{i=2}^{N} \left(1 - \frac{1}{K}\right)$$
$$= \frac{M}{s} \left(N - \frac{N-1}{K}\right).$$

Therefore, if $s < (1 - \frac{1}{K})M$ and $\frac{M}{s}\left(N - \frac{N-1}{K}\right) > M$, then there is no feasible federated schedule for task set $\tau_{counter}$ on M processors at such a speed s. □

For the example task set in Table 1, suppose that the speedup factor is $5 - \epsilon$ with $\epsilon > 0$. Then, we can conclude that τ_1 needs at least three processors and each task τ_i for $i = 2, 3, \ldots, 10$ needs exclusively at least two processors. Therefore, at least 21

processors are needed in this example task set under any federated scheduling with a speedup factor $5 - \epsilon$. The lower bound of Lemma 2 concludes that at least $\frac{10}{5-\epsilon}(10 - \frac{9}{2}) > 11$ processors are needed. Therefore, the speedup factor of any federated scheduling algorithm with respect to any optimal scheduling algorithm by considering this concrete example is at least 5. We now conclude the lower bound of the speedup factors of any federated scheduling algorithms with respect to any optimal scheduling algorithm.

Theorem 1. *The speedup factor of any federated scheduling algorithm with respect to any optimal scheduling algorithm for constrained-deadline task systems with DAG structures is at least $\min\left\{\left(1 - \frac{1}{K}\right)M, \left(N - \frac{N-1}{K}\right)\right\}$.*

Proof. By Lemma 1, task set $\tau_{counter}$ admits feasible schedules. By Lemma 2, if $s < (1 - \frac{1}{K})M$ and $\frac{M}{s}\left(N - \frac{N-1}{K}\right) > M$, then there is no feasible federated schedule for task set $\tau_{counter}$ on M processors at such a speed s. This implies that the resulting federated schedule under a speedup factor s with $s < (1 - \frac{1}{K})M$ and $s < N - \frac{N-1}{K}$ is not feasible on M processors. Therefore, for the task set $\tau_{counter}$, the speedup factor of any federated scheduling must be at least $\min\left\{\left(1 - \frac{1}{K}\right)M, \left(N - \frac{N-1}{K}\right)\right\}$. ☐

When K is 2, the speedup factor lower bound in Theorem 1 is at least $\min\{M/2, (N+1)/2\}$. As a conclusion, for the task set $\tau_{counter}$, any federated schedule has a speedup factor at least $\Omega\left(\min\{M, N\}\right)$ with respect to any optimal scheduling algorithm.

3 Conclusion and Discussions

The result in this paper shows that *at least in terms of the speedup metric with respect to any optimal scheduling algorithm, federated scheduling strategies do not yield any constant speedup factors for constrained-deadline task systems with DAG structures.* This also invalidates the conclusions of the algorithms in [1–3]:

> Baruah [1–3]: Our worst-case bounds indicate that at least in terms of the speedup metric, there is no loss in going from the three-parameter sporadic tasks model to the more general sporadic DAG tasks model.

That is, the above conclusions in [1–3] stated that the DAG structures (more precisely with the option of parallel executions) in addition to the traditional sporadic task model (by using only three parameters T_i, C_i, D_i for task τ_i with an assumption $C_i \leq D_i$) do not introduce additional penalty with respect to the speedup factors. The statement is only correct when the reference scheduling algorithm is the optimal federated scheduling algorithms. For the traditional sporadic task model without parallelism, there are scheduling algorithms with a constant speedup factor $3 - 1/M$ with respect to any optimal scheduling algorithm [5]. With the example provided in this paper, the above statement in [1–3] does not hold when we consider the speedup factors with respect to any optimal scheduling algorithm.

References

1. S. Baruah. The federated scheduling of constrained-deadline sporadic DAG task systems. In *Proceedings of the Design, Automation & Test in Europe Conference & Exhibition, DATE*, pages 1323–1328, 2015.

2. S. Baruah. Federated scheduling of sporadic DAG task systems. In *IEEE International Parallel and Distributed Processing Symposium, IPDPS*, pages 179–186, 2015.

3. S. Baruah. The federated scheduling of systems of conditional sporadic DAG tasks. In *Proceedings of the 15th International Conference on Embedded Software (EMSOFT)*, 2015.

4. S. K. Baruah, A. K. Mok, and L. E. Rosier. Preemptively scheduling hard-real-time sporadic tasks on one processor. In *IEEE Real-Time Systems Symposium*, pages 182–190, 1990.

5. J.-J. Chen and S. Chakraborty. Resource augmentation bounds for approximate demand bound functions. In *IEEE Real-Time Systems Symposium*, pages 272 – 281, 2011.

6. J. Li, J. Chen, K. Agrawal, C. Lu, C. D. Gill, and A. Saifullah. Analysis of federated and global scheduling for parallel real-time tasks. In *26th Euromicro Conference on Real-Time Systems, ECRTS*, pages 85–96, 2014.