Tardiness Bounds for Global EDF Scheduling on a Uniform Multiprocessor

Kecheng Yang and James H. Anderson
Department of Computer Science
University of North Carolina at Chapel Hill, USA
Email: {yangk, anderson}@cs.unc.edu

I. Introduction

Since the multicore revolution, the focus of research on real-time scheduling has shifted to multiprocessor platforms. *Global earliest-deadline-first (GEDF)* is a widely studied multiprocessor scheduling algorithm. Unfortunately, in hard real-time (HRT) systems where every deadline must be met, platform utilization must be restricted in order to guarantee schedulability under GEDF on a multiprocessor platform due to the Dhall Effect [2]. Nonetheless, in soft real-time (SRT) systems, where deadlines may be missed provided deadline tardiness is provably bounded, Devi and Anderson [1] showed the SRT-optimality of GEDF schedulers, *i.e.*, deadline tardiness bounds are guaranteed under GEDF scheduling for any feasible system.

The SRT-optimality result applies to both preemptive and non-preemptive GEDF schedulers, but only on a multiprocessor consisting of *identical* processors. The *uniform* multiprocessor model, which extends the identical multiprocessor model by associating a *speed* with each processor, is a widely studied heterogeneous multiprocessor model. We are interested in whether GEDF schedulers are still SRT-optimal on a uniform multiprocessor.

II. System Model

On a uniform multiprocessor platform, processors may have different speeds. The *speed* of processor p is denoted as s_p. Identical multiprocessors are a special case of uniform platforms, where all processors happen to have the same speed, which is usually normalized as 1.0. We consider the scheduling of n sequential sporadic tasks on a uniform multiprocessor platform of m processors. Each task τ_i is specified by (C_i, T_i), where C_i is its *worst-case execution requirement*, which is defined as its worst-case execution time on a unit-speed processor, and T_i is its *period*, or minimum separation between any two consecutive invocations of the task. Periodic tasks are a special case of sporadic tasks where the separation between any two consecutive invocations of a task τ_i is always *exact* T_i. We assume the deadlines to be *implicit* (*i.e.*, the relative deadline of each task equals its period).

A job is an invocation of a task. The j^{th} job of task τ_i is denoted as $\tau_{i,j}$ and is released at time $r_{i,j}$ with an absolute deadline $d_{i,j} = r_{i,j} + T_i$. If $\tau_{i,j}$ completes at time t_c, then its *tardiness* is defined as $\max\{0, t_c - d_{i,j}\}$. The tardiness of a task is the maximum tardiness of any of its jobs. A job is *ready* if and only if it is released and all previous jobs of the same task have already completed. Only ready jobs can commence execution since we assume each individual task to be sequential.

The utilization of a task τ_i is defined as $u_i = C_i/T_i$. Note that, on a uniform platform, $u_i \leq 1$ is not necessarily required for a system to be feasible. A feasibility condition, which depends on processor speeds, is provided by the following theorem cited from [3].

Theorem 1 (Theorem 4 in [3])**.** *Let S_k denote the sum of the k largest processor speeds, and let U_k denote the sum of the k largest task utilizations. Then, the n tasks can be scheduled to meet all deadlines on the m uniform processors, if and only if the following constraints hold:*

$$U_n \leq S_m, \tag{1}$$

$$U_k \leq S_k, \text{ for } k = 1, 2, \ldots, m-1. \tag{2}$$

The above feasibility condition was established for *periodic* tasks in *HRT* systems. Nonetheless, it is easy to show that (1) and (2) are also a necessary and sufficient feasibility condition for *sporadic* tasks in *SRT* systems.

Migrations. On an identical multiprocessor, a scheduler needs to determine *which* jobs are scheduled, but *where* they are scheduled matters less since every processor is the same. In contrast, on a uniform multiprocessor, *where* jobs are scheduled is crucial, and must be explicitly specified by a scheduler. In this context, we define that, under *non-preemptive* scheduling, once a job is scheduled on a particular processor, it continuously executes on that processor until complete.

III. Known Results and The Open Problem

Although both preemptive and non-preemptive GEDF schedulers are SRT-optimal on *identical* multiprocessors [1], it is shown in [6] that, on *uniform* platforms, there exists a feasible system where a task can become unboundedly tardy under

non-preemptive GEDF scheduling, which precludes the SRT-optimality of the non-preemptive GEDF scheduler. Therefore, we shift our focus to a fully preemptive GEDF scheduling algorithm **(A)** as follows.

(A) If at most m jobs are ready, then all ready jobs are scheduled; otherwise, the m ready jobs with earliest deadlines are scheduled. Migrations are allowed, by which the ready job with the k^{th} earliest deadline is always scheduled on the k^{th} fastest processor for any k. Deadline ties are broken arbitrarily.

Thus, we state the following open problem:

Under the preemptive GEDF scheduling algorithm **(A)***, is the deadline tardiness for every task provably bounded for any feasible system on a uniform multiprocessor?*

Some variants of this open problem have been solved in prior work, such as,

- relaxing the task model to allow intra-task parallelism [5];
- constraining per-task utilizations [4];
- restricting the platform to have only two uniform processors [6].

Therefore, to clarify, we would like to emphasize the following assumptions in this open problem:

- intra-task parallelism is strictly forbidden (jobs of the same task must execute in sequence);
- no constraints on task utilizations other than the feasibility condition (*i.e.*, (1) and (2)) are assumed;
- the uniform platform may have more than two processors.

To solve this open problem, we need to consider both potential answers. In order to answer "No" to this open problem, we need a counterexample where at least one task in a feasible system becomes unboundedly tardy under the preemptive GEDF scheduling algorithm **(A)**. The potential counterexample must have at least three uniform processors and likely has multiple high-utilization individual tasks (utilizations higher than the speeds of some processors in the system). On the other hand, in order to answer "Yes" to this open problem, we need to handle a key difference between an identical multiprocessor and a uniform one, as described in next section.

IV. The Key Difference

The difficulty in extending the SRT-optimality result on identical multiprocessors [1] to uniform multiprocessors comes from the change of feasibility condition. In the feasibility condition for scheduling a sporadic implicit-deadline task set on m identical processors, the *total utilization constraint*, which corresponds to (1), is that the sum of the utilizations of all tasks is at most m; the *per-task utilization constraint*, which corresponds to (2), is that the utilization of each individual task is at most 1.0. These two utilization constraints underlie the proof framework in [1].

In the feasibility condition for uniform platforms, (1) can play a similar role as the total utilization constraint in the identical case; however, (2) is not as convenient as the per-task utilization constraint in the identical case. Specifically, the per-task utilization constraint for an identical multiprocessor guarantees the following property.

(P) If any job $\tau_{i,j}$ continuously executes, it must complete within T_i time units, regardless of the processor on which $\tau_{i,j}$ executes.

This property is critical in the induction-based proof framework in [1], in order to handle the precedence constraint of each sequential sporadic task (*i.e.*, a job cannot commence execution until all previous jobs of the same task have already completed). However, **(P)** does not necessarily hold for a feasible system on a uniform multiprocessor.

Example 1. *We consider the scheduling of three sporadic tasks with utilizations $u_1 = u_2 = u_3 = 2$ on three uniform processors with speeds $s_1 = 4$ and $s_2 = s_3 = 1$. By checking* (1) *and* (2)*, this system is feasible. However,* **(P)** *does not hold when any job entirely executes on processor* 2 *or* 3*.*

Acknowledgment

This work is supported by NSF grants CPS 1239135, CNS 1409175, and CPS 1446631, AFOSR grant FA9550-14-1-0161, ARO grant W911NF-14-1-0499, and funding from General Motors.

References

[1] U. Devi and J. Anderson, "Tardiness bounds for global EDF scheduling on a multiprocessor," *Proceedings of the 26th IEEE Real-Time Systems Symposium*, 2005.

[2] S. Dhall and C. Liu, "On a real-time scheduling problem," *Operations Research*, 1978.

[3] S. Funk, J. Goossens, and S. Baruah, "On-line scheduling on uniform multiprocessors," *Proceedings of the 22nd IEEE Real-Time Systems Symposium*, 2001.

[4] G. Tong and C. Liu, "Supporting soft real-time sporadic task systems on heterogeneous multiprocessors with no uilitzation loss," *IEEE Transactions on Parallel and Distributed Systems*, 2015.

[5] K. Yang and J. Anderson, "Optimal GEDF-based schedulers that allow intra-task parallelism on heterogeneous multiprocessors," *Proceedings of the 12th IEEE Symposium on Embedded Systems for Real-Time Multimedia*, 2014.

[6] K. Yang and J. Anderson, "On the soft real-time optimality of global EDF on multiprocessors: From identical to uniform heterogeneous," *Proceedings of the 21st IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, 2015.