The Journal of Systems and Software 89 (2014) 99–108

Contents lists available at ScienceDirect

The Journal of Systems and Software

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

Demand-based schedulability analysis for real-time multi-core scheduling

Jinkyu Lee [a], Insik Shin [b,*]

[a] Department of Computer Science and Engineering, Sungkyunkwan University, South Korea
[b] Department of Computer Science, KAIST, Daejeon, South Korea

ARTICLE INFO

Article history:
Received 20 January 2013
Received in revised form 30 July 2013
Accepted 19 September 2013
Available online 24 October 2013

Keywords:
Demand-based schedulability analysis
Real-time systems
Multi-core systems

ABSTRACT

In real-time systems, schedulability analysis has been widely studied to provide offline guarantees on temporal correctness, producing many analysis methods. The demand-based schedulability analysis method has a great potential for high schedulability performance and broad applicability. However, such a potential is not yet fully realized for real-time multi-core scheduling mainly due to (i) the difficulty of calculating the resource demand under dynamic priority scheduling algorithms that are favorable to multi-cores, and (ii) the lack of understanding how to combine the analysis framework with deadline-miss conditions specialized for those scheduling algorithms. Addressing those two issues, to the best of our knowledge, this paper presents the first demand-based schedulability analysis for dynamic job-priority scheduling algorithms: EDZL (Earliest Deadline first until Zero-Laxity) and LLF (Least Laxity First), which are known to be effective for real-time multi-core scheduling. To this end, we first derive demand bound functions that compute the maximum possible amount of resource demand of jobs of each task while the priority of each job can change dynamically under EDZL and LLF. Then, we develop demand-based schedulability analyses for EDZL and LLF, by incorporating those new demand bound functions into the existing demand-based analysis framework. Finally, we combine the framework with additional deadline-miss conditions specialized for those two laxity-based dynamic job-priority scheduling algorithms, yielding tighter schedulability analyses. Via simulations, we demonstrate that the proposed schedulability analyses outperform the existing schedulability analyses for EDZL and LLF.

1. Introduction

In the area of real-time systems in which temporal correctness is critical, real-time scheduling has been substantially studied to support real-time tasks without violating their own timing constraints. Those studies generally focus on two important issues: scheduling algorithms that determine the order of execution of a set of tasks, and schedulability analysis that verifies the temporal correctness of the task set under a specific scheduling algorithm. For uniprocessor systems with a sporadic task model (Mok, 1983), it has been proved that EDF (Earliest Deadline First) (Liu and Layland, 1973) is an optimal scheduling algorithm, and its demand-based schedulability analysis (Baruah et al., 1990) captures the optimality of EDF, i.e., the analysis is sufficient and necessary.

As multi-core architectures become popular due to its high computing capability with low power consumption, real-time multi-core scheduling has been paid an increasing amount of attention; many scheduling algorithms and their schedulability analyses have been adapted from real-time uniprocessor scheduling theories and/or newly developed. However, schedulability analysis methods still have much room for improvement in that most existing multi-core schedulability analyses are only sufficient, not necessary.

For tighter schedulability analysis on multi-cores, we focus on the demand-based schedulability analysis method (Baruah, 2007). Different from other existing analysis methods, such as utilization-based (Goossens et al., 2003), deadline-based (Bertogna et al., 2009) and response-time-based schedulability analysis methods (Bertogna and Cirinei, 2007), the demand-based method investigates a large number of intervals with different length, and calculates the resource requirements that a task set collectively *demands* in each interval to satisfy their individual timing constraints. Then, a task set is deemed schedulable if for all the intervals, resources can be supplied to the task set more than or equal to its resource *demand*. Hence, the demand-based method has a great potential for broad applicability and high schedulability performance. That is, once the resource demand of a task set is computed under a specific scheduling algorithm, we can easily apply the demand-based schedulability analysis framework to the algorithm. Also, the method can reduce the pessimism of

* Corresponding author. Tel.: +82 42 350 3524.
E-mail addresses: jinkyu.yi@gmail.com (J. Lee), insik.shin@cs.kaist.ac.kr (I. Shin).

0164-1212/$ – see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.jss.2013.09.029

calculating the effect of *carry-in* jobs[1] of a task set in a given interval, by limiting the contribution of carry-in jobs through the comprehensive interval check. As a result of such a tight calculation, the demand-based method yields the exact schedulability analysis of EDF on a uniprocessor platform (Baruah et al., 1990) and one of the best EDF schedulability analyses on a multi-core platform (Baruah, 2007) (See a survey Bertogna and Baruah, 2011).

While the demand-based schedulability analysis method has achieved a great success for some simple scheduling algorithms (e.g., EDF), its potential has not been fully realized for more dynamic scheduling algorithms that are suitable for multi-cores, mainly because of two reasons. First, it is challenging to calculate the amount of resource demand under those algorithms. Second, it has not been tried to combine the demand-based schedulability analysis method with additional deadline-miss conditions specialized for those algorithms, which potentially improves schedulability.

Addressing the two issues, this paper develops demand-based schedulability analyses for two dynamic job-priority scheduling algorithms: EDZL (Earliest Deadline first until Zero Laxity) (Lee, 1994) and LLF (Least Laxity First) (Leung, 1989). EDZL is a modification of EDF; it gives the highest priority to zero-laxity jobs, and schedules other jobs according EDF, where a laxity of a job at any time instant is defined as the remaining time to its deadline minus the remaining execution time at the instant. LLF prioritizes jobs according to their laxity values; the lower laxity, the higher priority. EDZL and LLF are not only proven as optimal on a uniprocessor platform (Dertouzos and Mok, 1989; Park et al., 2005) as well as EDF, but also known to be effective on a multiprocessor platform (Cho et al., 2002; Lee et al., 2010). This is because the algorithms promote zero-laxity jobs, which should be scheduled immediately to avoid deadline misses.

To develop demand-based schedulability analyses for EDZL and LLF, we need to calculate the resource demand under the algorithms. Since the priority of jobs dynamically changes under EDZL and LLF, we investigate how long a job of a later deadline can interfere another job of an earlier deadline under EDZL and LLF. Based on this, we derive demand bound functions under EDZL and LLF, and then we develop two types of schedulability analyses using the functions. First, we simply apply the functions to the existing demand-based schedulability analysis framework. Second, we incorporate the framework into additional deadline-miss conditions specialized for the laxity-based dynamic change of job priority, resulting in tighter schedulability analyses.

We demonstrate through simulation that the proposed demand-based schedulability analyses not only outperform existing schedulability analyses for EDZL (Baker et al., 2008; Lee and Shin, 2013) and LLF (Lee et al., 2010, 2012), but also find a large number of additional schedulable task sets, which are not covered by the existing schedulability analyses. As a bonus, if we apply our analyses to uniprocessors, they become exact schedulability analyses of EDZL and LLF, providing an alternative way to prove the optimality of EDZL and LLF on a uniprocessor platform.

In summary, this paper makes the following contributions.

- It calculates demand bound functions for EDZL and LLF, under which the priority of jobs dynamically changes. To the best knowledge of the authors, this study presents the first demand bound functions for dynamic job-priority scheduling algorithms;
- It develops demand-based schedulability analyses for EDZL and LLF by incorporating the functions and the existing demand-based schedulability analysis framework into additional deadline-miss conditions specialized for laxity-based dynamic priority change, thus broadening applicability of the demand-based schedulability analysis method; and
- It demonstrates the effectiveness of the proposed demand-based schedulability analyses through simulation, showing that they improve the state-of-the-art analysis methods.

The remainder of this paper is organized as follows. Section 2 introduces our system model, assumptions and notations, and then summarizes the existing demand-based schedulability analysis for EDF. Section 3 derives demand bound functions under EDZL and LLF. Section 4 develops demand-based schedulability analyses for EDZL and LLF using the derived functions. Section 5 evaluates the schedulability performance of the analyses. Section 6 discusses related work, and finally Section 7 concludes this paper.

2. Background

In this section, we first present our system model, assumptions and notations. Then, we recapitulate the existing demand-based schedulability analysis for EDF (Baruah, 2007), which will be a basis for our demand-based schedulability analyses for EDZL and LLF.

2.1. System model, assumptions and notations

In this paper we assume a sporadic task model (Mok, 1983), in which a task $\tau_i \in \tau$ is specified by (T_i, C_i, D_i), where T_i is the minimum separation, C_i is the worst-case execution time, and D_i is the relative deadline. Further, we restrict our attention to implicit ($C_i \le D_i = T_i$) and constrained ($C_i \le D_i \le T_i$) deadline tasks. Let $|\tau|$ denote the number of tasks in a task set τ. A task τ_i invokes a series of jobs, each separated from its predecessor by at least T_i amount of time, and supposed to finish its execution within D_i amount of time. We assume that a single job of a task cannot be executed in parallel.

We focus on a multi-core system, in which there are m identical cores. Also, we deal with global, preemptive scheduling algorithms under which jobs can be executed in any core (global), and a higher-priority job can preempt another low-priority executing job any time (preemptive). Without loss of generality, let one time unit denote the quantum length, and therefore all task parameters are specified as multiples of the quantum length.

To express a job in an interval, we use the following terms: in an interval $[t_a, t_b]$, a job is called

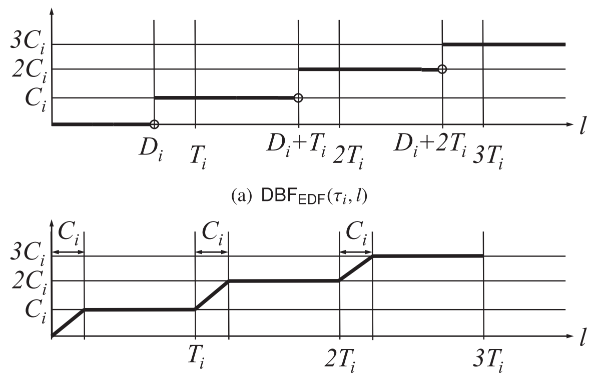
- *carry-in*, if the job is released before t_a and has remaining execution at t_a;
- *carry-out*, if the job is released before t_b and has remaining execution at t_b; and
- *body*, if the release time and deadline of the job are within the interval $[t_a, t_b]$.

Also, we express that a job J_1 *interferes* another job J_2 in a time slot, if J_1 executes but J_2, although ready to execute, does not execute in the time slot.

2.2. Existing demand-based schedulability analysis for EDF

The notion of resource demand has been successfully employed in many schedulability analyses on various platforms (Baruah et al., 1990; Shin and Lee, 2003; Baruah, 2007). The demand of a given interval represents the amount of execution that should be performed in order to finish the body jobs without any deadline miss. A task set is schedulable under a scheduling algorithm if the demand of each interval is no greater than the amount of resource supplied within the same interval. A key issue is then how to calculate the demand of any given interval accurately, and it requires a good

[1] A job is said to be a carry-in job in an interval, if it is released before the interval and has remaining execution at the beginning of the interval.



(a) $\mathrm{DBF}_{\mathrm{EDF}}(\tau_i, l)$

(b) $\mathrm{DBF}'_{\mathrm{EDF}}(\tau_i, l)$

Fig. 1. Demand bound functions for EDF.

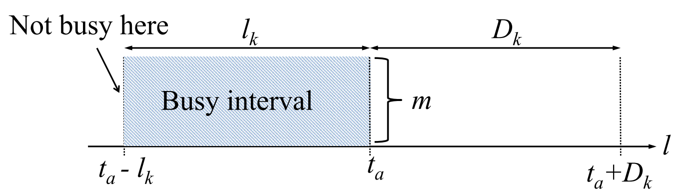


Fig. 2. Demand-based analysis framework.

understanding of which jobs would execute within the interval, according to the priorities assigned to the jobs.

Under EDF, job priorities are assigned according to deadlines, and carry-out jobs do not interfere the execution of body jobs. This makes it possible to calculate the demand of an interval without considering carry-out jobs. Considering this property, Baruah et al. (1990) introduced a demand bound function of a task τ_i for an interval of length l, which computes the demand as the cumulative maximum execution requirements of τ_i' *body jobs only* (denoted by $\mathrm{DBF}_{\mathrm{EDF}}(\tau_i, l)$) as follows.

$$\mathrm{DBF}_{\mathrm{EDF}}(\tau_i, l) \triangleq \left(\left\lfloor \frac{l - D_i}{T_i} \right\rfloor + 1 \right) \cdot C_i. \quad (1)$$

In Baruah (2007), a demand bound function of a task τ_i for an interval of length l, which computes the demand as the cumulative maximum execution requirements of τ_i' *body jobs and a carry-in job* (denoted by $\mathrm{DBF}'_{\mathrm{EDF}}(\tau_i, l)$) is calculated as follows.

$$\mathrm{DBF}'_{\mathrm{EDF}}(\tau_i, l) \triangleq \left\lfloor \frac{l}{T_i} \right\rfloor \cdot C_i + \min(C_i, l \bmod T_i). \quad (2)$$

The functions $\mathrm{DBF}_{\mathrm{EDF}}(\tau_i, l)$ and $\mathrm{DBF}'_{\mathrm{EDF}}(\tau_i, l)$ are illustrated in Fig. 1. Note that it trivially holds that $\mathrm{DBF}'_{\mathrm{EDF}}(\tau_i, l) \geq \mathrm{DBF}_{\mathrm{EDF}}(\tau_i, l)$ for all $l \geq 0$ and $\tau_i \in \tau$.

Using the demand bound functions, the demand-based schedulability analysis for EDF on a multi-core platform has been developed in Baruah (2007), by identifying necessary deadline miss conditions of each task. To do this, the analysis performs the following steps.

1 It focuses on a job of a task τ_k whose release time and deadline are t_a and $t_a + D_k$, respectively.
2 It considers an interval $[t_a - l_k, t_a + D_k)$ such that $[t_a - l_k, t_a)$ is the maximum consecutive busy interval (in which all cores are occupied), and at least one core is idle in $[t_a - l_k - 1, t_a - l_k)$. See Fig. 2.
3 It calculates the demand of all jobs in $[t_a - l_k, t_a + D_k)$, except the job of interest. Here at most $m - 1$ tasks have their carry-in job because at least one core is idle in $[t_a - l_k - 1, t_a - l_k)$ by the definition of the maximum busy interval of $[t_a - l_k, t_a)$.
4 If the amount of demand is equal to or larger than $m \cdot (l_k + D_k - C_k + 1)$, we deem τ_k unschedulable.
5 If Step 4 does not deem τ_k unschedulable for all $l_k \geq 0$, we finally guarantee that any job of τ_k does not miss its deadline.

The above steps are mathematically expressed as follows.

Lemma 1. *(Theorem 2 in Baruah, 2007) A task set τ is schedulable under EDF on an m-core platform, if every task $\tau_k \in \tau$ satisfies the following condition for all $l_k \geq 0$[2]:*

$$\sum_{\tau_i \in \tau} I_{\mathrm{EDF}}(\tau_i, l_k) + \sum_{m-1 \text{ largest } \tau_i \in \tau} \{ I'_{\mathrm{EDF}}(\tau_i, l_k) - I_{\mathrm{EDF}}(\tau_i, l_k) \} < m \cdot (l_k + D_k - C_k + 1), \quad (3)$$

where

$$I_{\mathrm{EDF}}(\tau_i, l) = \begin{cases} \min(\mathrm{DBF}_{\mathrm{EDF}}(\tau_i, l + D_k), l + D_k - C_k + 1), & \text{if } \tau_i \neq \tau_k, \\ \min(\mathrm{DBF}_{\mathrm{EDF}}(\tau_i, l + D_k) - C_k, l), & \text{if } \tau_i = \tau_k, \end{cases} \quad (4)$$

and

$$I'_{\mathrm{EDF}}(\tau_i, l) = \begin{cases} \min(\mathrm{DBF}'_{\mathrm{EDF}}(\tau_i, l + D_k), l + D_k - C_k + 1), & \text{if } \tau_i \neq \tau_k, \\ \min(\mathrm{DBF}'_{\mathrm{EDF}}(\tau_i, l + D_k) - C_k, l), & \text{if } \tau_i = \tau_k. \end{cases} \quad (5)$$

Proof. The proof is given in Baruah (2007). Also, this proof is similar to the demand-based schedulability analyses for EDZL and LLF to be presented in Theorem 1 in Section 4. □

Note that Lemma 1 is slightly different from Theorem 2 in Baruah (2007); while the latter employed continuous time, the former employed discrete time, which has been widely applied to Bertogna and Cirinei (2007), Bertogna et al. (2009), and Lee et al. (2010, 2012).

Then, Lemma 1 is a generalization of the exact analysis of EDF on a uniprocessor platform (Baruah et al., 1990) in that they are equivalent when $m = 1$ (Baruah, 2007).

Building upon Lemma 1 (Baruah, 2007), the demand-based schedulability analysis method has been applied to FP (Fixed Priority) scheduling (Guan et al., 2009). For FP, it has been proven that $l_k = 0$ is the critical instant (Davis and Burns, 2011), so we do not need to investigate $l_k > 0$.

Although such a demand-based analysis method in Lemma 1 is potentially applicable to any other scheduling algorithms, it has not been adapted to any dynamic job-priority scheduling algorithms (e.g., EDZL and LLF) that are more suitable for multi-core scheduling. This raises two key challenges. First, how can we calculate demand bound functions of a task set while their job priorities are dynamically changing? Second, how can we incorporate the demand bound functions into deadline-miss conditions specialized for those algorithms? In the next sections, we develop demand-based schedulability analyses for EDZL and LLF, by addressing those two challenges.

3. Demand bound functions under EDZL and LLF

In this section, we calculate the resource demand under EDZL and LLF. To do this, we first upper-bound how long a job with a later deadline can interfere another job with an earlier deadline. Then, we derive the functions under EDZL and LLF, which will be key components for demand-based schedulability analyses for EDZL and LLF to be presented in Section 4.

[2] An upper-bound of l_k is given in Baruah (2007).

3.1. Demand bound functions under EDZL

EDZL assigns the highest priority to jobs with the zero laxity state, and prioritizes other jobs according to EDF. Unlike EDF, a carry-out job can interfere a body-job if the carry-out job encounters the zero-laxity state. Therefore, a challenge for deriving a demand bound function under EDZL is to calculate the amount of execution that a carry-out job of each task demands. The following lemma is useful to address the challenge.

Lemma 2. *Under EDZL, a job J_i of τ_i with a later deadline $t_0+\alpha$ $(\alpha>0)$ can interfere another job J_k of τ_k with an earlier deadline t_0 during at most $C_i-\alpha$ amount of time.*

Proof. Since the deadline of J_i is later than that of J_k, J_i can interfere J_k only when it encounters the zero-laxity state. The earliest time instant for J_i to have a zero laxity is at $t_0+\alpha-C_i$; this happens when J_i does not perform any execution until it encounters the zero-laxity state. Therefore, J_i can interfere J_k only in $[t_0+\alpha-C_i,\ t_0)$. Since J_k's deadline is t_0, the amount of time J_i interferes J_k is at most $C_i-\alpha$ time units. □

By Lemma 2, a job of τ_i with a deadline of $t_0+\alpha$ demands at most $C_i-\alpha$ amount of execution before t_0. Then, such a demand of a carry-out job is equal to zero when α is equal to or larger than C_i, and it linearly increases up to C_i when α decreases to zero.

Then, we will consider two cases for the maximum demand of jobs of τ_i in an interval, depending on whether there is a carry-in job or not. We first look at the case of no carry-in job. Fig. 3(a) represents the release pattern of jobs of τ_i that maximizes the demand of jobs of τ_i under EDZL in case of no carry-in job. In this pattern, the release time of the first body job of τ_i is aligned with the beginning of the interval of interest. Then, the demand of body jobs of τ_i is the same as the EDF case, i.e., $\mathsf{DBF}_{\mathsf{EDF}}(\tau_i, l)$. By Lemma 2, the demand of a carry-out job is positive only when the difference between the end of the interval of interest (i.e., t_0) and the deadline of the carry-out job (i.e., $t_0+\alpha$) is less than C_i. If we express this mathematically, the demand of a carry-out job is positive only when $D_i-C_i+T_i\cdot x<l<D_i+T_i\cdot x$, where $x=0, 1, 2, \ldots$, and here the amount of the demand is $l-(D_i-C_i+T_i\cdot x)$; otherwise, there is no carry-out job or there is no resource demand by a carry-out job. Then, considering this worst-case release pattern, Fig. 4(a) illustrates the demand bound function of τ_i under EDZL in an interval of length l in case of no carry-in job (denoted by $\mathsf{DBF}_{\mathsf{EDZL}}(\tau_i, l)$), which is mathematically expressed as follows:

$$\mathsf{DBF}_{\mathsf{EDZL}}(\tau_i, l) \triangleq \mathsf{DBF}_{\mathsf{EDF}}(\tau_i, l) + \max\left(0, l-\left(\left\lfloor\frac{l-D_i}{T_i}\right\rfloor+1\right)\cdot T_i-(D_i-C_i)\right). \quad (6)$$

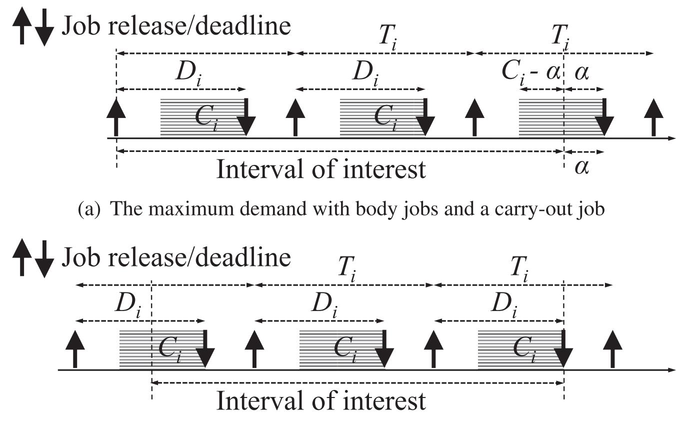
In case of allowing a carry-in job, Fig. 3(b) illustrates the worst-case release pattern of jobs of τ_i that maximizes the demand of jobs of τ_i. Here, the last body job's deadline is aligned with the end of the interval of interest. Therefore, there exists no carry-out job of τ_i that can contribute to the demand in this worst-case release pattern. This means, the demand bound function of τ_i under EDZL in an interval of length l in case of allowing a carry-in job (denoted by $\mathsf{DBF}'_{\mathsf{EDZL}}(\tau_i, l)$) is the same as that under EDF as follows:

$$\mathsf{DBF}'_{\mathsf{EDZL}}(\tau_i, l) \triangleq \mathsf{DBF}'_{\mathsf{EDF}}(\tau_i, l), \quad (7)$$

where Fig. 4(b) represents $\mathsf{DBF}'_{\mathsf{EDZL}}(\tau_i, l)$.

3.2. Demand bound functions under LLF

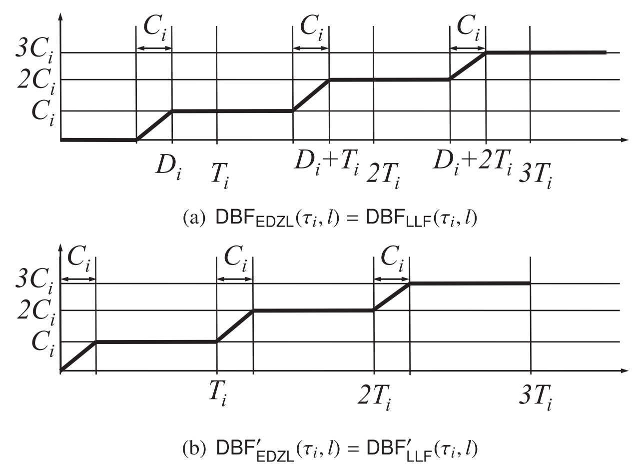
LLF assigns priorities based on laxity values: the smaller laxity, the higher priority. From the laxity-based priority assignment, we present three properties of LLF scheduling as follows.



(a) The maximum demand with body jobs and a carry-out job

(b) The maximum demand with any job

Fig. 3. The release patterns of jobs of τ_i that maximize the demand in an interval under EDZL and LLF.



(a) $\mathsf{DBF}_{\mathsf{EDZL}}(\tau_i, l) = \mathsf{DBF}_{\mathsf{LLF}}(\tau_i, l)$

(b) $\mathsf{DBF}'_{\mathsf{EDZL}}(\tau_i, l) = \mathsf{DBF}'_{\mathsf{LLF}}(\tau_i, l)$

Fig. 4. Demand bound functions for EDZL and LLF.

Observation 1. *LLF has the following properties:*

1. *The laxity of a job will decrease by one if the job does not execute in the current time slot, or remain the same otherwise.*
2. *A job J_1 can interfere another job J_2 only when the laxity of J_1 is equal to or smaller than that of J_2.*
3. *If both J_1 and J_2 have the same laxity at t, the two jobs have the same priority. Then, J_1 can interfere J_2 at most the amount of remaining execution of J_2 from t on. This is because, after J_1 interferes J_2 in a time slot, the priority of J_1 will become lower than that of J_2 by the first property of LLF. Then, J_1 cannot interfere J_2 until J_2 performs at least one execution (while J_1 does not perform the execution). Therefore, if the laxity of J_1 at t is the same as that of J_2, J_1 can interfere J_2 at most the amount of remaining execution of J_2 at t.*

Using the above three properties of LLF, we now derive demand bound functions under LLF. To calculate this, we present the same lemma as EDZL. Note that, although the statement of the lemma for LLF is coincidentally the same as Lemma 2 for EDZL, the proof is totally different and more complicated.

Lemma 3. *Under LLF, a job J_i of τ_i with a later deadline $t_0+\alpha$ $(\alpha>0)$ can interfere another job J_k of τ_k with an earlier deadline t_0 during at most $C_i-\alpha$ amount of time.*

Proof. For notational convenience, let $C_i(t)$ and $C_k(t)$ denote the remaining execution time of J_i and J_k at t, and let $L_i(t)$ and $L_k(t)$ denote the laxity of J_i and J_k at t.

For J_i to interfere J_k, the laxity of J_i should be no more than that of J_k. We consider two cases: (a) the laxity of J_i is always strictly smaller than that of J_k in $[t_0-\beta, t_0)$ where $t_0-\beta$ ($\beta>0$) is the later time instant between the release times of J_i and J_k; and (b) there exists the earliest time instant $t_0-\gamma$ ($\gamma>0$) at which the laxity of J_i is equal to that of J_k, i.e., $L_k(t_0-\gamma)=L_i(t_0-\gamma)$.

Case (a): here we assume $C_k(t_0-\beta)>0$; otherwise, J_k already finishes its execution before $t_0-\beta$, and therefore J_i cannot interfere J_k. We show the contradiction of the assumption, if J_i interferes J_k during more than $C_i-\alpha$ amount of time.

Suppose that J_i interferes J_k during more than $C_i-\alpha$ time units in $[t_0-\beta, t_0)$. According to the first and second properties in Observation 1, for J_i to have a smaller laxity than J_k after interfering during more than $C_i-\alpha$ time units, it holds that $L_i(t_0-\beta)<L_k(t_0-\beta)-(C_i-\alpha)$. Since the remaining times to the deadlines of J_i and J_k at $t_0-\beta$ are $\beta+\alpha$ and β, respectively, the following inequality holds by the definition of laxity:

$$\begin{aligned} & L_i(t_0-\beta) < L_k(t_0-\beta) - C_i + \alpha \\ & \Leftrightarrow \beta+\alpha - C_i(t_0-\beta) < \beta - C_k(t_0-\beta) - C_i + \alpha \\ & \Leftrightarrow C_k(t_0-\beta) < C_i(t_0-\beta) - C_i \\ & \Rightarrow C_k(t_0-\beta) \le 0. \end{aligned} \tag{8}$$

This is a contradiction of $C_k(t_0-\beta)>0$.

Case (b): by definition, J_k has $C_k(t_0-\gamma)$ amount of remaining execution at $t_0-\gamma$, and its remaining time to deadline is γ, resulting in $L_k(t_0-\gamma)=\gamma-C_k(t_0-\gamma)$. Since $L_k(t_0-\gamma)=L_i(t_0-\gamma)$ and J_i has $\gamma+\alpha$ amount of remaining time to deadline at $t_0-\gamma$, the amount of remaining execution of J_i at $t_0-\gamma$ is calculated by $(\gamma+\alpha)-(\gamma-C_k(t_0-\gamma))=\alpha+C_k(t_0-\gamma)$. By the third property of Observation 1, J_i interferes J_k at most $C_k(t_0-\gamma)$ amount of time in $[t_0-\gamma, t_0)$ So, among $\alpha+C_k(t_0-\gamma)$ amount remaining execution, J_i interferes only during at most $C_k(t_0-\gamma)$, which means J_i cannot interfere J_k during at least $\alpha+C_k(t_0-\gamma)-C_k(t_0-\gamma)=\alpha$ amount of time. Considering the initial execution time of J_i is C_i, we conclude that J_i interferes J_k during at most $C_i-\alpha$ amount of time.

The cases (a) and (b) prove the lemma. □

Since Lemma 3 is coincidentally the same as Lemma 2, we can consider the same release patterns that maximize the resource demand of jobs of a given task as shown in Fig. 3. Therefore, demand bound functions of τ_i under LLF in an interval of length l in case of no carry-in job (denoted by $\mathsf{DBF}_{\mathsf{LLF}}(\tau_i, l)$) and the existence of a carry-in job (denoted by $\mathsf{DBF}'_{\mathsf{LLF}}(\tau_i, l)$) are the same as those under EDZL as follows:

$$\mathsf{DBF}_{\mathsf{LLF}}(\tau_i, l) \triangleq \mathsf{DBF}_{\mathsf{EDZL}}(\tau_i, l), \tag{9}$$

$$\mathsf{DBF}'_{\mathsf{LLF}}(\tau_i, l) \triangleq \mathsf{DBF}'_{\mathsf{EDZL}}(\tau_i, l), \tag{10}$$

where the functions are also illustrated in Fig. 4.

4. Demand-based schedulability analysis of EDZL and LLF

In this section, we develop demand-based schedulability analysis of EDZL and LLF. To do this, we first apply the derived demand bound functions to the existing demand-based schedulability analysis framework. Then, we incorporate the framework into additional deadline-miss conditions specialized for EDZL and LLF, yielding tighter schedulability analyses.

4.1. Schedulability analysis of EDZL and LLF with deadline-miss conditions

By applying the derived demand bound functions under EDZL and LLF to the existing demand-based schedulability analysis framework in Lemma 1, we develop demand-based schedulability analyses for EDZL and LLF, as stated in the following theorem.

Theorem 1. *A task set τ_i is schedulable under EDZL and LLF on an m-core platform, if every task $\tau_k \in \tau$ satisfies the following condition for all $l_k \ge 0$*[3]*:*

$$\sum_{\tau_i\in\tau} I_{\mathsf{EDZL}}(\tau_i, l_k) + \sum_{m-1 \text{ largest } \tau_i\in\tau} \{I'_{\mathsf{EDZL}}(\tau_i, l_k) - I_{\mathsf{EDZL}}(\tau_i, l_k)\} < m\cdot(l_k+D_k-C_k+1), \tag{11}$$

where

$$I_{\mathsf{EDZL}}(\tau_i, l) = \begin{cases} \min(\mathsf{DBF}_{\mathsf{EDZL}}(\tau_i, l+D_k), l+D_k-C_k+1), & \text{if } \tau_i \ne \tau_k, \\ \min(\mathsf{DBF}_{\mathsf{EDZL}}(\tau_i, l+D_k)-C_k, l), & \text{if } \tau_i = \tau_k, \end{cases} \tag{12}$$

and

$$I'_{\mathsf{EDZL}}(\tau_i, l) = \begin{cases} \min(\mathsf{DBF}'_{\mathsf{EDZL}}(\tau_i, l+D_k), l+D_k-C_k+1), & \text{if } \tau_i \ne \tau_k, \\ \min(\mathsf{DBF}'_{\mathsf{EDZL}}(\tau_i, l+D_k)-C_k, l), & \text{if } \tau_i = \tau_k. \end{cases} \tag{13}$$

Proof. For given $l_k=l$, we want to judge whether or not a job of τ_k (whose release time and deadline are t_a and t_a+D_k) misses its deadline, provided that an interval $[t_a-l, t_a)$ is the maximum busy interval, as shown in Fig. 2. Then, for the job of interest to miss its deadline, the amount of the resource demand of all jobs except the job of interest in $[t_a, t_a+D_k)$ should be at least $m\cdot(D_k-C_k+1)$; otherwise, C_k amount of execution of the job of interest will be successfully performed before t_a+D_k, regardless of execution patterns of other jobs. Considering that $[t_a-l, t_a)$ is a busy interval, the amount of the resource demand of all jobs except the job of interest in $[t_a-l, t_a+D_k)$ should be at least $m\cdot(l+D_k-C_k+1)$.

Then, the remaining step is the LHS of Eq. (11) is a safe upper-bound of such a resource demand. Similar to Baruah (2007), let Γ_k denote a set of intervals (not necessarily continuous) of length D_k-C_k+1 over $[t_a, t_a+D_k)$; Γ_k is chosen such that the amount of execution of other jobs than the job of interest is maximized. Since the length of Γ_k is D_k-C_k+1, the amount of actual demand of jobs of τ_i in Γ_k should be upper-bounded by D_k-C_k+1, which is followed by the fact that the amount of actual demand of jobs of τ_i in $[t_a-l, t_a)\cup\Gamma_k$ should be upper-bounded by $l+D_k-C_k+1$. This is why $I_{\mathsf{EDZL}}(\tau_i, l)$ and $I'_{\mathsf{EDZL}}(\tau_i, l)$ are upper-bounded by $l+D_k-C_k+1$ in Eqs. (12) and (13) for $\tau_i \ne \tau_k$.

In case of $\tau_i=\tau_k$, there is no demand in $[t_a, t_a+D_k)$ because we do not count the job of interest of τ_k. Therefore, we deduct C_k amount from the demand, and upper-bound the demand by l (i.e., the length of $[t_a-l, t_a)$) as shown in Eqs. (12) and (13) for $\tau_i=\tau_k$.

Then, $I_{\mathsf{EDZL}}(\tau_i, l)$ and $I'_{\mathsf{EDZL}}(\tau_i, l)$ in Eqs. (12) and (13) successfully upper-bound the amount of resource demand of jobs of τ_i in $[t_a-l, t_a)\cup\Gamma_k$, in case of no carry-in job and the existence of a carry-in job, respectively. By the definition of the maximum busy interval, we know that at most $m-1$ tasks have their carry-in jobs in $[t_a-l, t_a+D_k)$, but we do not know which tasks have their carry-in jobs. To safely upper-bound the demand by considering $I_{\mathsf{EDZL}}(\tau_i, l) \le I'_{\mathsf{EDZL}}(\tau_i, l)$ for all $\tau_i\in\tau$ and $l\ge 0$, we initially add $I_{\mathsf{EDZL}}(\tau_i, l)$ for $\tau_k\in\tau$. Then, we add the difference between $I'_{\mathsf{EDZL}}(\tau_i, l)$ and $I_{\mathsf{EDZL}}(\tau_i, l)$ for $m-1$ tasks, which have the largest difference. Then, under any combination with up to $m-1$ tasks having their carry-in jobs, we can safely upper-bound the amount of the demand.

Therefore, the LHS of Eq. (11) is a safe upper-bound of the resource demand of all jobs except the job of interest in $[t_a-l,$

[3] An upper-bound of l_k will be derived in Section 4.3.

$t_a) \cup \Gamma_k$. This means, if Eq. (11) holds for given l, a job of τ_k cannot miss its deadline, provided that $[t_a - l, t_a)$ is the maximum busy interval. Since we test Eq. (11) for all $l \geq 0$, the theorem holds. □

Then, Theorem 1 can be an exact schedulability of EDZL and LLF on a uniprocessor platform, as stated in the following lemma.

Lemma 4. *Theorem 1 provides an alternative way to prove that EDZL and LLF are optimal scheduling algorithms on a uniprocessor platform.*

Proof. In Baruah et al. (1990), it has been proved that a scheduling algorithm is optimal on a uniprocessor platform, if the algorithm makes every task set τ schedulable, which satisfies for all $l > 0$:

$$\sum_{\tau_i \in \tau} \mathrm{DBF}_{\mathrm{EDF}}(\tau_i, l) \leq l. \tag{14}$$

If we apply $m = 1$ to Theorem 1, the schedulability analysis is reduced as follows: a task set τ is schedulable, if the following inequality holds for all $l \geq \min_{\tau_i \in \tau} D_i$:

$$\sum_{\tau_i \in \tau} \mathrm{DBF}_{\mathrm{EDZL}}(\tau_i, l) \leq l, \tag{15}$$

which is equivalent to the above optimality condition on a uniprocessor platform. □

While it is well-known that EDZL and LLF outperform EDF on a multiprocessor platform (Baker et al., 2008; Lee et al., 2010), the schedulability analysis of EDZL and LLF in Theorem 1, although it generalizes the exact schedulability analysis on uniprocessors, cannot be better than the schedulability analysis of EDF in Lemma 1. This is because $\mathrm{DBF}_{\mathrm{EDZL}}(\tau_i, l) \geq \mathrm{DBF}_{\mathrm{EDF}}(\tau_i, l)$ and $\mathrm{DBF}'_{\mathrm{EDZL}}(\tau_i, l) = \mathrm{DBF}'_{\mathrm{EDF}}(\tau_i, l)$ hold for any $\tau_i \in \tau$ and $l > 0$.

Therefore, we now derive another demand-based schedulability analysis of EDZL and LLF, by incorporating additional deadline-miss conditions specialized for EDZL and LLF, which will be detailed in the next subsection.

4.2. Schedulability analysis of EDZL and LLF with zero-laxity conditions

Under any zero-laxity-based algorithm, which assigns the highest priority to zero-laxity jobs, such as EDZL and LLF, additional deadline-miss conditions have been identified (Baker et al., 2008; Lee et al., 2011) as stated in the following observation.

Observation 2. *Under any zero-laxity-based algorithm, for a job to miss its deadline, there should be $m+1$ zero-laxity jobs before the deadline miss.*

Since zero-laxity jobs have the highest priority, a zero-laxity job misses its deadline only when m other zero-laxity jobs are executed.

Using the observation, we can develop another schedulability analysis by checking whether a task set can have $m+1$ tasks whose job can enter the zero-laxity state, which has been combined with other schedulability analysis methods (Baker et al., 2008; Lee and Shin, 2013). Now, we incorporate the observation to the demand-based schedulability analysis.

Theorem 2. *A task set τ_i is schedulable under EDZL and LLF on an m-core platform, if at least $|\tau| - m$ tasks $\tau_k \in \tau$ satisfy the following condition for all $l_k \geq 0$[4]:*

$$\sum_{\tau_i \in \tau} \hat{I}_{\mathrm{EDZL}}(\tau_i, l_k) + \sum_{m-1 \text{ largest } \tau_i \in \tau} \{\hat{I}'_{\mathrm{EDZL}}(\tau_i, l_k) - \hat{I}_{\mathrm{EDZL}}(\tau_i, l_k)\} < m \cdot (l_k + D_k - C_k), \tag{16}$$

where

$$\hat{I}(\tau_i, l) = \begin{cases} \min(\mathrm{DBF}_{\mathrm{EDZL}}(\tau_i, l + D_k), l + D_k - C_k), & \text{if } \tau_i \neq \tau_k, \\ \min(\mathrm{DBF}_{\mathrm{EDZL}}(\tau_i, l + D_k) - C_k, l), & \text{if } \tau_i = \tau_k, \end{cases} \tag{17}$$

and

$$\hat{I}'(\tau_i, l) = \begin{cases} \min(\mathrm{DBF}'_{\mathrm{EDZL}}(\tau_i, l + D_k), l + D_k - C_k), & \text{if } \tau_i \neq \tau_k, \\ \min(\mathrm{DBF}'_{\mathrm{EDZL}}(\tau_i, l + D_k) - C_k, l), & \text{if } \tau_i = \tau_k. \end{cases} \tag{18}$$

Proof. The proof is similar to Theorem 1. We focus on $l_k = l$ such that $[t_a - l, t_a)$ is the maximum busy interval in which m cores are occupied in $[t_a - l, t_a)$ but at least one core is idle in $[t_a - l - 1, t_a - l)$.

Consider a job of τ_k, whose release time and deadline are t_a and $t_a + D_k$, respectively. Similar to the definition of Γ_k, let Γ'_k denote a set of intervals (not necessarily continuous) of length $D_k - C_k$ over $[t_a, t_a + D_k)$; Γ'_k is also chosen such that the amount of executions of other jobs than the job of interest is maximized. Then, for the job of interest of τ_k to encounter the zero-laxity state, the resource demand of all other jobs than the job of interest in Γ'_k should at least $m \cdot (D_k - C_k)$; otherwise, the job of interest finishes C_k amount of execution without entering the zero-laxity state. Considering that $[t_a - l, t_a)$ is the maximum busy interval, the resource demand of all other jobs than the job of interest in $[t_a - l, t_a) \cup \Gamma'_k$ should be at least $m \cdot (l + D_k - C_k)$.

Then, since the length of Γ'_k is $D_k - C_k$, $\hat{I}_{\mathrm{EDZL}}(\tau_i, l)$ for $\tau_i \neq \tau_k$ in Eq. (17), which means the amount of the resource demand of jobs of τ_i in $[t_a - l, t_a) \cup \Gamma'_k$, should be upper-bounded by $l + D_k - C_k$. The same upper-bound holds for $\hat{I}'_{\mathrm{EDZL}}(\tau_i, l)$ in Eq. (18). On the other hand, it holds that $\hat{I}_{\mathrm{EDZL}}(\tau_i, l) = I_{\mathrm{EDZL}}(\tau_i, l)$ and $\hat{I}'_{\mathrm{EDZL}}(\tau_i, l) = I'_{\mathrm{EDZL}}(\tau_i, l)$ for $\tau_i = \tau_k$, because they are irrelevant to Γ_k and Γ'_k.

By testing Eq. (16) for all $l \geq 0$, we can guarantee that a job of τ_k cannot encounter the zero-laxity state. Then, by Observation 2, we guarantee that a task set τ is schedulable if at least $|\tau| - m$ tasks' jobs cannot reach the zero-laxity state, which proves the theorem. □

Note that the schedulability analyses of EDZL and LLF in Theorems 1 and 2 are sustainable with respect to the minimum separations, the worst-case execution times, and the relative deadlines (i.e., $\{T_i, C_i, D_i\}_{\tau_i \in \tau}$) (Burns and Baruah, 2008), which means a task set deemed schedulable by the analyses is also deemed schedulable even if T_i and/or D_i increase and/or C_i decreases.

4.3. Time-complexity

To make Theorems 1 and 2 practical, we need to upper-bound l_k in the theorems; otherwise, it is intractable to test the theorems. Using the idea from Baruah (2007), we now calculate an upper-bound of l_k for Theorems 1 and 2, as stated in the following lemma.

[4] An upper-bound of l_k will be derived in Section 4.3.

Lemma 5. *For Theorems 1 and 2, it is safe to investigate only a set of l_k that satisfies the following inequality:*

$$l_k \leq \frac{\text{numerator}}{m - \sum_{\tau_i \in \tau}(C_i/T_i)}, \tag{19}$$

where $\text{numerator} = \sum_{\tau_i \in \tau} C_i - m \cdot D_k + m \cdot C_k + D_k \cdot \sum_{\tau_i \in \tau}(C_i/T_i) + \sum_{\tau_i \in \tau}(T_i - D_i) \cdot (C_i/T_i)$.

Proof. It is easily checked that $I_{\mathsf{EDZL}}(\tau_i, l_k), I'_{\mathsf{EDZL}}(\tau_i, l_k), \hat{I}_{\mathsf{EDZL}}(\tau_i, l_k)$ and $\hat{I}'_{\mathsf{EDZL}}(\tau_i, l_k)$ in Eqs. (12), (13), (17) and (18) are upper-bounded by $\mathsf{DBF}_{\mathsf{EDF}}(\tau_i, l_k + D_k) + C_k$. Therefore, to violate Eq. (11) or (16), the following inequality holds necessarily:

$$\sum_{\tau_i \in \tau} C_i + \sum_{\tau_i \in \tau} \mathsf{DBF}_{\mathsf{EDF}}(\tau_i, l_k + D_k) \geq m \cdot (l_k + D_k - C_k). \tag{20}$$

Using the technique in Baruah et al. (1990), i.e., $\mathsf{DBF}_{\mathsf{EDF}}(\tau_i, l) \leq l \cdot \frac{C_i}{T_i} + (T_i - D_i) \cdot \frac{C_i}{T_i}$, the following inequality holds.

Eq. (20)

$$\Rightarrow \sum_{\tau_i \in \tau} C_i + (l_k + D_k) \cdot \sum_{\tau_i \in \tau} \frac{C_i}{T_i} + \sum_{\tau_i \in \tau}(T_i - D_i) \cdot \frac{C_i}{T_i} \geq m \cdot (l_k + D_k - C_k)$$

$$\Leftrightarrow l_k \leq \frac{\text{numerator}}{m - \sum_{\tau_i \in \tau}(C_i/T_i)}. \tag{21}$$

□

Then, the time-complexity of Theorems 1 and 2 with Lemma 5 is pseudo-polynomial in the task parameters, if the task set τ satisfies that $\sum_{\tau_i \in \tau}(C_i/T_i)$ is upper-bounded by a constant strictly smaller than m.

It is tractable to apply schedulability analyses with pseudo-polynomial time-complexity offline (see such analyses in Bertogna and Baruah, 2011). Therefore, schedulability analyses in Theorems 1 and 2 can provide offline guarantees on temporal correctness of a given task set scheduled by EDZL or LLF.

5. Evaluation

In this section, we compare schedulability performance of the proposed analyses with existing analyses.

5.1. Task set generation

We generate task sets based on a popular technique (Baker, 2005) employed in many studies, e.g., Bertogna et al. (2009) and Lee et al. (2012). We have three input parameters: (a) the number of cores m (2, 4, 8 or 16), (b) the type of tasks (constrained or implicit deadline tasks), and (c) individual task utilization (C_i/T_i) distribution—bimodal with parameters: 0.1, 0.3, 0.5, 0.7 or 0.9, or exponential with parameters: 0.1, 0.3, 0.5, 0.7 or 0.9. Here, for a given bimodal parameter p, a value for C_i/T_i is uniformly chosen in $[0, 0.5)$ with probability p, and in $[0.5, 1)$ with probability $1 - p$. Also, for a given exponential parameter $1/\lambda$, a value for C_i/T_i is selected according to the exponential distribution whose probability density function is $\lambda \cdot \exp(-\lambda \cdot x)$.

For each task, T_i is uniformly distributed in $[1, 1000]$, C_i is chosen based on (c), and D_i is uniformly chosen in $[C_i, T_i]$ for constrained deadline task sets or D_i is set to T_i for implicit deadline task sets.

For each combination of (a), (b) and (c), we repeat the following steps and generate 100,000 task sets.

1. We generate $m+1$ tasks.
2. In order to exclude unschedulable sets, we check whether the current task set can pass a necessary feasibility condition (Baker and Cirinei, 2006).
3. If it fails to pass the feasibility test, we discard the current task set and return to Step 1. Otherwise, we include the current set for evaluation. Then, this set serves as a basis for the next new task set; we create a new set by adding a new task into the current set and return to Step 2.

Therefore, the number of the first task set is $m+1$ while that of the second, third, ... task sets is $m+2$, $m+3$, ... as long as each task set passes the necessary feasibility condition (Baker and Cirinei, 2006). If the condition is not satisfied, the number of tasks in the next task set is $m+1$, and we repeat the task set increment. This procedure enables to generate a wide variety of task sets in terms of the number of tasks.

For any given m and given type of task sets, 100,000 task sets are created for each task utilization model, thus yielding 1,000,000 task sets in total.

5.2. Schedulability performance

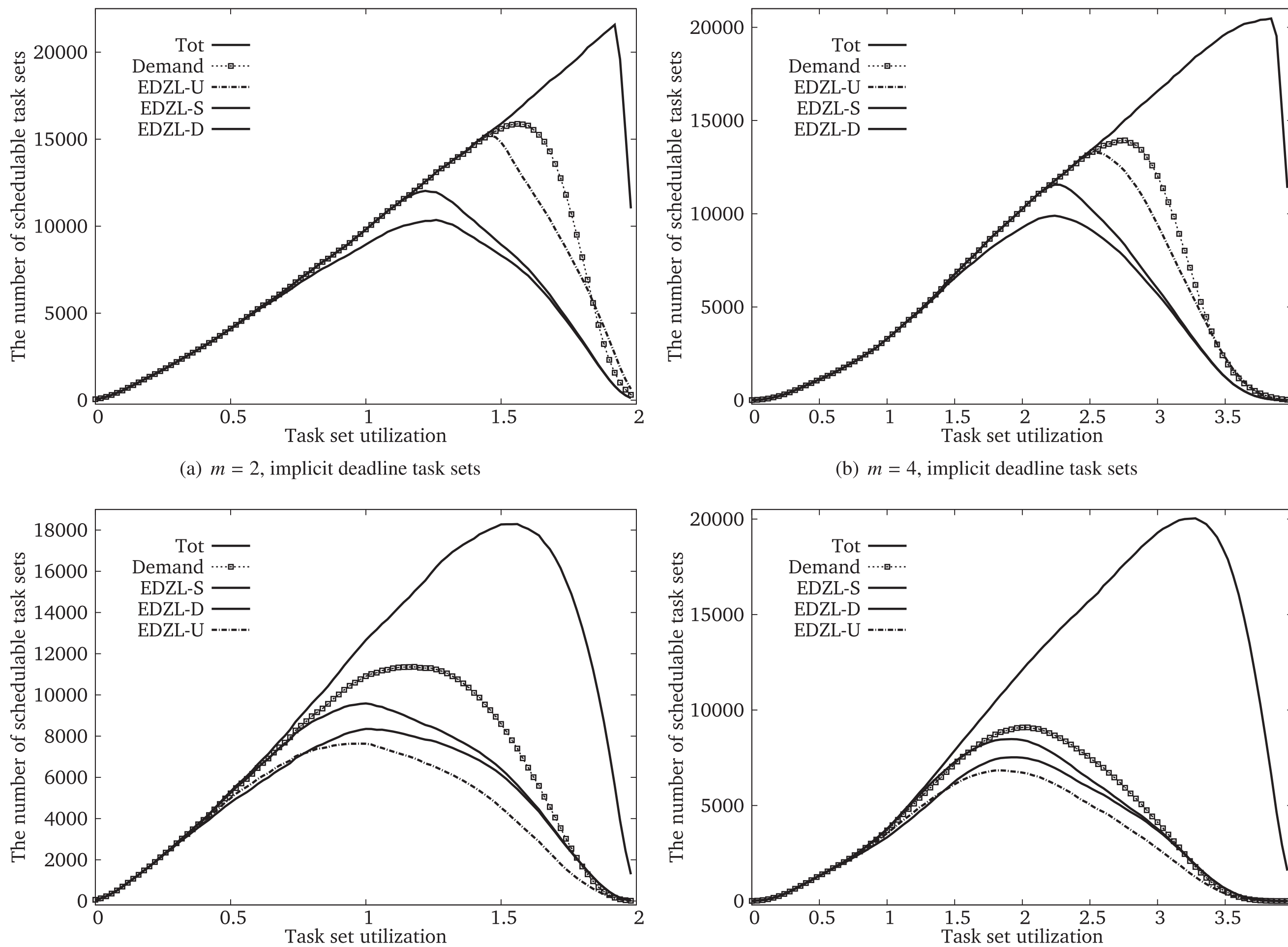
We compare the following schedulability analyses for EDZL and LLF:

- Our demand-based schedulability analysis of EDZL and LLF, i.e., schedulable by either Theorem 1 or 2 (denoted by Demand),
- An existing deadline-based EDZL schedulability analysis, i.e., schedulable by Theorem 7 in Baker et al. (2008) (denoted by EDZL-D),
- An existing slack-based EDZL schedulability analysis, i.e., schedulable by Section 9 in Baker et al. (2008) (denoted by EDZL-S),
- An existing utilization-based EDZL schedulability analysis, i.e., schedulable by Theorem 2 in Lee and Shin (2013) (denoted by EDZL-U),
- An existing deadline-based LLF schedulability analysis, i.e., schedulable by Theorem 3 in Lee et al. (2010) (denoted by LLF-D), and
- An existing slack-based LLF schedulability analysis, i.e., schedulable by Fig. 5 in Lee et al. (2012) (denoted by LLF-S).

In Figs. 5 and 6, each plot shows the number of task sets proven schedulable by each test, with task set utilization ($\sum_{\tau_i \in \tau} C_i/T_i$) in $[\sum_{\tau_i \in \tau} C_i/T_i - 0.01 \cdot m, \sum_{\tau_i \in \tau} C_i/T_i + 0.01 \cdot m)$. In the figures, "Tot" represents the number of generated task sets.

We first compare EDZL schedulability analyses. As shown in Fig. 5, EDZL-U and EDZL-S are better than other existing EDZL schedulability analyses, respectively for implicit and constrained deadline task sets. For example, if we focus in Fig. 5(a) (i.e., $m=2$, implicit deadline task sets), EDZL-U deems 74.1% task sets schedulable while EDZL-D and EDZL-S prove only 55.9% and 60.2% task sets schedulable. Likewise, EDZL-D, EDZL-S, and EDZL-U respectively deem 48.4%, 53.3%, and 43.1% constrained task sets schedulable when $m=2$, (i.e., Fig. 5(c)). However, Demand outperforms all existing EDZL schedulability analyses, with $m=2$ and 4. For example, Demand deems 77.9% (61.6%) implicit (constrained) task sets schedulable when $m=2$, which are 3.8%–22.0% (8.3%–18.5%) improvement compared to existing EDZL schedulability analyses. Also, Demand covers up to 4.0% additional schedulable task sets with $m=8$ and 16, which are deemed schedulable by neither EDZL-D, EDZL-S nor EDZL-U.

When it comes to LLF schedulability analyses, Demand is (one of) the best LLF schedulability analysis as shown in Fig. 6. That is, while LLF-D and LLF-S prove that 39.8%–63.5% and 43.3%–65.4% task sets



(a) m = 2, implicit deadline task sets

(b) m = 4, implicit deadline task sets

(c) m = 2, constrained deadline task sets

(d) m = 4, constrained deadline task sets

Fig. 5. Schedulability performance of EDZL schedulability analyses.

are schedulable with $m=2$ and 4, Demand deems 41.8%–77.9% task sets schedulable under LLF. For $m=8$ and $m=16$, Demand finds up to 5.2% additional schedulable task sets, which are not covered by both LLF-D and LLF-S.

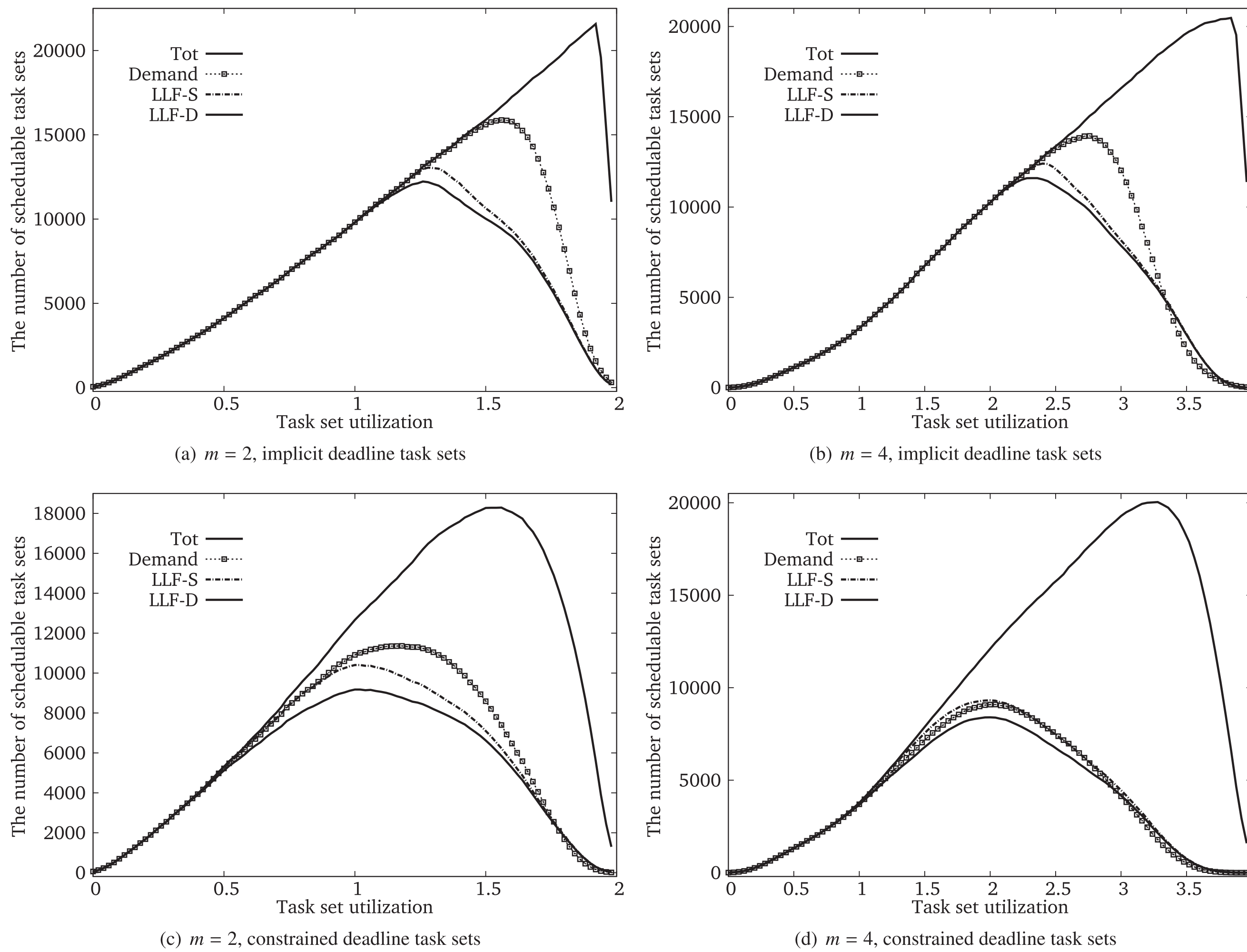
Here an interesting point is that Demand does not outperform LLF-D and LLF-S under some environment, e.g., Fig. 6(d) with $m=4$ and constrained deadline task sets. This is because, Demand only utilizes zero-laxity conditions for tighter analysis. However, under LLF, there are additional deadline-miss conditions that are connected with complex laxity dynamics; the schedulability difference between EDZL-D and LLF-D, or EDZL-S and LLF-S comes from those conditions. Therefore, one may improve the demand-based schedulability analysis of LLF by incorporating those conditions into Demand. Note that those conditions have been presented in Lee et al. (2010, 2012), but the conditions make schedulability analyses remarkably complicated. Therefore, the issue is beyond the scope of this paper, and we will briefly discuss this in Section 7. Also, since LLF may incur frequent job preemptions, we need to consider such overhead. Section 6 in Lee et al. (2012) has introduced a variant of LLF to explore the tradeoff between the overhead and schedulability improvement, and it would be interesting to extend the demand-based analysis to the variant, which is also beyond the scope of this paper.

When it comes to time-complexity, EDZL-D, EDZL-U, EDZL-S, LLF-D and LLF-S have $|\tau|^2$, $m \cdot |\tau|$, $|\tau|^3 \cdot \max_{\tau_i \in \tau} D_i$, $|\tau|^2 \cdot \max_{\tau_i \in \tau} C_i \cdot (D_i - C_i)$, and $|\tau|^2 \cdot \max_{\tau_i \in \tau} D_i \cdot \sum_{\tau_i \in \tau} (D_i - C_i)$, respectively (Lee and Shin, 2013; Lee et al., 2012). In particular, the time-complexity of the schedulability analyses with higher schedulability performance (i.e., EDZL-S and LLF-S) is pseudo-polynomial in the task parameters, which is the same as Ours. As shown in Table 1, in any case, the actual running time of Demand for each implicit deadline task set does not exceed 1 second, implying that Demand is a tractable offline schedulability test.

In summary, our demand-based schedulability analyses for EDZL and LLF not only become (one of) the best single schedulability analysis for EDZL and LLF, but also cover a large number of additional schedulable task sets by EDZL and LLF, which are not proven by existing schedulability analyses.

Table 1
Average running time of implicit deadline task sets

	$m=2$	$m=4$	$m=8$	$m=16$
EDZL-D	<1 ms	<1 ms	<1 ms	<1 ms
EDZL-S	<1 ms	<1 ms	<1 ms	<1 ms
EDZL-U	<1 ms	<1 ms	<1 ms	<1 ms
LLF-D	1.6 ms	4.9 ms	15.4 ms	57.0 ms
LLF-S	2.1 ms	7.2 ms	25.6 ms	109.5 ms
Demand	2.0 ms	7.6 ms	30.6 ms	127.7 ms



(a) $m = 2$, implicit deadline task sets

(b) $m = 4$, implicit deadline task sets

(c) $m = 2$, constrained deadline task sets

(d) $m = 4$, constrained deadline task sets

Fig. 6. Schedulability performance of LLF schedulability analyses.

6. Related work

To guarantee the schedulability of a task set under a specific scheduling algorithm, schedulability analysis has been widely studied. For uniprocessors, utilization-based schedulability analyses of EDF and FP (Liu and Layland, 1973) have been proposed for implicit deadline task sets. Since the analyses are no longer effective for constrained deadline task sets, people have proposed more sophisticated methods. As a result, the demand-based schedulability analysis method yields the exact schedulability analyses of EDF (Baruah et al., 1990), while the response-time-based method does that of FP (Audsley et al., 1991). With increasing popularity of multi-core chips, people have tried to adapt the two effective methods to real-time multi-core scheduling. The response-time-based method (Bertogna and Cirinei, 2007) and its simpler versions (the deadline-based and slack-based methods, Bertogna et al., 2009) have yielded many schedulability analyses for existing scheduling algorithms, e.g., Bertogna and Cirinei (2007) and Bertogna et al. (2009) for EDF, Bertogna and Cirinei (2007), Bertogna et al. (2009), and Guan et al. (2009) for FP, Baker et al. (2008) and Lee and Shin (2013) for EDZL, Lee et al. (2012) for LLF, Davis and Burns (2011) for FPZL (Fixed Priority until Zero-Laxity), Chwa et al. (2012) for SPDF (Smallest Pseudo Deadline First), etc.

However, when it comes to the demand-based schedulability analysis method, such adaptation for multi-cores has been realized only for EDF (Baruah, 2007) and FP (Guan et al., 2009) scheduling[5]. Since the demand-based schedulability analysis of EDF is one of the best among a large number of existing EDF analyses in terms of schedulability performance (See a survey Bertogna and Baruah, 2011), we expect that such superiority of the demand-based method would hold for other scheduling algorithms that have been proven effective for real-time multi-core scheduling, e.g., EDZL and LLF. As we mentioned in the previous sections, there are two challenges to develop demand-based schedulability analyses of those scheduling algorithms: how to compute demand bound functions of those algorithms, and how to incorporate the demand-based method into deadline miss conditions specialized for those algorithms. Addressing the two challenges, this paper developed demand-based schedulability analyses of EDZL and LLF, and demonstrated their effectiveness in terms of schedulability performance.

7. Conclusion

In this paper, we have addressed two issues that block the practical applicability of the demand-based schedulability analysis method for real-time multi-core scheduling: (i) the way of

[5] For FP, it has been proved that the demand-based schedulability analysis can be reduced to the deadline-based analysis (Guan et al., 2009; Davis and Burns, 2011) as we mentioned in Section 2.

calculating the resource demand under dynamic, complex algorithms, and (ii) the way of incorporating the analysis method into additional deadline-miss conditions specialized for those algorithms. As a result, the demand-based schedulability analyses for EDZL and LLF outperform existing schedulability analyses.

One direction of future work is to apply the demand-based schedulability analysis framework to other scheduling algorithms than zero-laxity-based scheduling. Also, it would be interesting to incorporate the framework into additional laxity conditions for LLF. To do this, it is necessary to apply complex laxity dynamics, and to develop a large number of additional demand bound functions that calculate the resource demand of other jobs that have higher priority than a job of interest with *certain laxity* at *certain time instant*. Another direction is to extend our work to dependent task models. For example, it would be interesting to develop scheduling algorithms and their demand-based schedulability analyses by addressing cache interference among tasks (Yan and Zhang, 2011; Ding and Zhang, 2013; Guan et al., 2009).

Acknowledgements

This work was supported in part by BRL (2009-0086964), BSRP (2012-R1A1A1014930), NCRC (2012-0000980), KEIT (2011-10041313), and KIAT (M002300089) funded by the Korea Government (MEST/MSIP/MOTIE).

References

Audsley, N., Burns, A., Richardson, M., Wellings, A., 1991. Hard real-time scheduling: the deadline-monotonic approach. In: Proceedings of the IEEE Workshop on Real-Time Operating Systems and Software, pp. 133–137.
Baker, T., 2005. An analysis of EDF schedulability on a multiprocessor. IEEE Transactions on Parallel and Distributed Systems 16 (8), 760–768.
Baker, T.P., Cirinei, M., 2006. A necessary and sometimes sufficient condition for the feasibility of sets of sporadic hard-deadline tasks. In: Proceedings of IEEE Real-Time Systems Symposium, pp. 178–190.
Baker, T.P., Cirinei, M., Bertogna, M., 2008. EDZL scheduling analysis. Real-Time Systems 40, 264–289.
Baruah, S., Mok, A., Rosier, L., 1990. Preemptively scheduling hard-real-time sporadic tasks on one processor. In: Proceedings of IEEE Real-Time Systems Symposium, pp. 182–190.
Baruah, S., 2007. Techniques for multiprocessor global schedulability analysis. In: Proceedings of IEEE Real-Time Systems Symposium, pp. 119–128.
Bertogna, M., Cirinei, M., Lipari, G., 2009. Schedulability analysis of global scheduling algorithms on multiprocessor platforms. IEEE Transactions on Parallel and Distributed Systems 20, 553–566.
Bertogna, M., Cirinei, M., 2007. Response-time analysis for globally scheduled symmetric multiprocessor platforms. In: Proceedings of IEEE Real-Time Systems Symposium.
Bertogna, M., Baruah, S., 2011. Tests for global EDF schedulability analysis. Journal of Systems Architecture 57 (5), 487–497.
Burns, A., Baruah, S., 2008. Sustainability in real-time scheduling. Journal of Computing Science and Engineering 2 (1), 74–97.
Cho, S., Lee, S.-K., Ahn, S., Lin, K.-J., 2002. Efficient real-time scheduling algorithms for multiprocessor systems. IEICE Transaction on Communications E85-B (12), 2859–2867.
Chwa, H.S., Back, H., Chen, S., Lee, J., Easwaran, A., Shin, I., Lee, I., 2012. Extending task-level to job-level fixed priority assignment and schedulability analysis using pseudo-deadlines. In: Proceedings of IEEE Real-Time Systems Symposium, pp. 51–62.
Davis, R.I., Burns, A., 2011. FPZL schedulability analysis. In: Proceedings of IEEE Real-Time Technology and Applications Symposium, pp. 245–256.
Davis, R., Burns, A., 2011. Improved priority assignment for global fixed priority pre-emptive scheduling in multiprocessor real-time systems. Real-Time Systems 47, 1–40.
Dertouzos, M.L., Mok, A.K., 1989. Multiprocessor on-line scheduling of hard-real-time tasks. IEEE Transactions on Software Engineering 15, 1497–1506.
Ding, Y., Zhang, W., 2013. Multicore real-time scheduling to reduce inter-thread cache interferences. Journal of Computing Science and Engineering 7 (1), 67–80.
Goossens, J., Funk, S., Baruah, S., 2003. Priority-driven scheduling of periodic task systems on multiprocessors. Real-Time Systems 25 (2-3), 187–205.
Guan, N., Stigge, M., Yi, W., Yu, G., 2009. Cache-aware scheduling and analysis for multicores. In: Proceedings of ACM & IEEE International Conference on Embedded Software, pp. 245–254.
Guan, N., Sitgge, M., Yi, W., Yu, G., 2009. New response time bounds for fixed priority multiprocessor scheduling. In: Proceedings of IEEE Real-Time Systems Symposium, pp. 387–397.
Lee, S.K., 1994. On-line multiprocessor scheduling algorithms for real-time tasks. In: IEEE Region 10's Ninth Annual International Conference, pp. 607–611.
Lee, J., Easwaran, A., Shin, I., 2010. LLF schedulability analysis on multiprocessor platforms. In: Proceedings of IEEE Real-Time Systems Symposium, pp. 25–36.
Lee, J., Shin, I., 2013. EDZL schedulability analysis in real-time multi-core scheduling. IEEE Transactions on Software Engineering 39 (7), 910–916.
Lee, J., Easwaran, A., Shin, I., 2012. Laxity dynamic and LLF schedulability analysis on multiprocessor platforms. Real-Time Systems (6), 716–749.
Lee, J., Easwaran, A., Shin, I., Lee, I., 2011. Zero-laxity based real-time multiprocessor scheduling. Journal of Systems and Software 84 (12), 2324–2333.
Leung, J.Y.-T., 1989. A new algorithm for scheduling periodic, real-time tasks. Algorithmica 4, 209–219.
Liu, C., Layland, J., 1973. Scheduling algorithms for multi-programming in a hard-real-time environment. Journal of the ACM 20 (1), 46–61.
Mok, A., 1983. Fundamental design problems of distributed systems for the hard-real-time environment, Ph.D. thesis. Massachusetts Institute of Technology.
Park, M., Han, S., Kim, H., Cho, S., Cho, Y., 2005. Comparison of deadline-based scheduling algorithms for periodic real-time tasks on multiprocessor. IEICE Transaction on Information and Systems E88-D, 658–661.
Shin, I., Lee, I., 2003. Periodic resource model for compositional real-time guarantees. In: Proceedings of IEEE Real-Time Systems Symposium, pp. 2–13.
Yan, J., Zhang, W., 2011. Bounding worst-case performance for multi-core processors with shared L2 instruction caches. Journal of Computing Science and Engineering 5 (1), 1–18.

Jinkyu Lee received B.S., M.S. and Ph.D. degrees in Computer Science in 2004, 2006 and 2011, respectively, from KAIST (Korea Advanced Institute of Science and Technology), South Korea. Since October 2011, he is a research fellow/visiting scholar in Department of Electrical Engineering and Computer Science, University of Michigan, USA. His research interests include system design and analysis with timing guarantees, QoS support, and resource management in real-time embedded systems and cyber-physical systems. He won the best student paper award from the 17th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS) in 2011, and the best paper award from the 33rd IEEE Real-Time Systems Symposium (RTSS) in 2012.

Insik Shin is currently an associate professor in Dept. of Computer Science at KAIST, South Korea, where he joined in 2008. He received a B.S. from Korea University, an M.S. from Stanford University, and a Ph.D. from University of Pennsylvania all in Computer Science in 1994, 1998, and 2006, respectively. He has been a post-doctoral research fellow at Malardalen University, Sweden, and a visiting scholar at University of Illinois, Urbana-Champaign until 2008. His research interests lie in cyber-physical systems and real-time embedded systems. He is currently a member of Editorial Boards of Journal of Computing Science and Engineering. He has been co-chairs of various workshops including satellite workshops of RTSS, CPSWeek and RTCSA and has served various program committees in real-time embedded systems, including RTSS, RTAS, ECRTS, and EMSOFT. He received best paper awards, including Best Paper Awards from RTSS in 2003 and 2012, Best Student Paper Award from RTAS in 2011, and Best Paper runner-ups at ECRTS and RTSS in 2008.