Reserving Processors by Precise Scheduling of Mixed-Criticality Tasks

Tianning She  
Texas State University  
t_s374@txstate.edu

Zhishan Guo  
University of Central Florida  
zguo@ucf.edu

Qijun Gu  
Texas State University  
qijun@txstate.edu

Kecheng Yang  
Texas State University  
yangk@txstate.edu

Abstract—Mixed-criticality (MC) scheduling has been proposed to mitigate the pessimism in real-time schedulability analysis that must provide guarantees for the worst case. In most existing work on MC scheduling, low-critical tasks are either dropped or degraded at the criticality mode switch in order to preserve the temporal guarantees for high-critical tasks. Recently, a different direction, called precise MC scheduling, has been investigated. In precise MC scheduling, no low-critical task should be dropped or degraded; instead, the platform processing capacity is augmented at mode switch to accommodate the additional workload by high-critical tasks. In contrast to prior work on this topic with respect to varying processor speed, this work investigates the precise scheduling problem of MC tasks when the number of available processors may vary at the mode switch. To address this new problem, we propose two alternative algorithms by adapting virtual-deadline-based EDF and by fluid scheduling, respectively, and provide a sufficient schedulability test for each. We also conduct schedulability experiments with randomly generated task sets to demonstrate the effectiveness of the proposed algorithms and the benefits of the new scheduling model.

Index Terms—mixed-criticality tasks, precise scheduling, reserving processors, virtual deadlines, fluid scheduling.

I. INTRODUCTION

To mitigate the pessimism in real-time schedulability analysis that must provide guarantees for the worst case, mixed-criticality (MC) scheduling has been proposed, featured by modeling a single system parameter with multiple estimates. In particular, two estimates on the worst-case execution time (WCET) of a task are the most commonly studied in the literature, where two system modes are defined depending on which of the two estimates are respected.

A significant body of existing research on MC scheduling was conducted in the direction that low-critical tasks are sacrificed during execution, completely or partially, in the worst case in order to provide the real-time guarantees to high-critical tasks. However, Ernst and Di Natale suggested that dropping or degrading service, even if for low-critical (rather than non-critical) tasks, may be infeasible or problematic for some applications [10]. Thus, the precise MC scheduling has been proposed [6], where even low-critical tasks are guaranteed full service. In precise MC scheduling, the system mode switch concerns degrading the active processors in typical scenarios for energy conservation while enabling the full capability of the underlying platform to preserve the real-time guarantees in the worst-case scenarios.

Energy efficiency is essential, especially for embedded systems, which often rely on unreliable energy sources such as batteries or energy harvesting devices. To improve energy efficiency, many modern processors are equipped with a feature, called dynamic voltage and frequency scaling (DVFS), which enables dynamic adjustment of processor voltage and frequency, i.e., the speed of processors may vary during runtime. However, DVFS has a major limitation: it is not effective in reducing static/leakage power consumption, which may elevate to 50% or more of the overall power consumption [12]. By contrast, dynamic power management (DPM) and deep sleep modes can lead to significant energy conservation resulted from the power-down of a number of system components—not only the cores but also their associated caches, translation look-aside buffers, etc. Similar to DVFS yielding varying-speed processors, DPM and deep sleep modes may cause the number of active processors to vary as well.

In this paper, we consider a new precise MC scheduling problem inspired by DPM and deep sleep modes. We interpret the two WCET estimates of an MC task as a typical-case and a worst-case upper bounds on its execution time. While the worst-case bound shall never be exceeded, whether the actual execution time during runtime will respect the typical-case bound or not is unknown prior to runtime. If all actual execution times are indeed within their typical-case bounds, then the tasks are executed only on a subset of the processors, leaving other processors inactive and in the deep sleep mode by DPM. Once it is observed that any actual execution time exceeds its typical-case bound, all processors should be immediately activated to ensure that all deadlines are still met. As suggested by its name, the typical-case upper bound should be respected in most of the time and the worst-case mode should be a rare event. Therefore, the proposed precise scheduling of MC tasks could significantly reduce the system energy consumption while preserving the real-time guarantees for all tasks even in the worst case.

Related Work. Since it was introduced by Vestal [19], MC tasks and their scheduling have attracted a huge amount of interest in the real-time systems research community. (Please see [8] for a comprehensive survey on this topic.) Initially, most works were directed to scenarios where all low-critical tasks

This work is supported in part by NSF grant CNS-1850851, a start-up grant from the University of Central Florida, and start-up and REP grants from Texas State University.
are completely dropped if any high-critical task behaves its worst case. More recently, this over-sacrificing was criticized, and gradual degradation of low-critical tasks was investigated. To provide degraded service, the imprecise MC model [7] was proposed, where the execution of low-critical tasks is reduced but not dropped even in the worst case. Several subsequent works [4, 7, 11, 13, 14, 17] explored various definitions of this execution reduction. To eliminate such reduction, the problem of precise MC scheduling was proposed and investigated on varying-speed uniprocessors [6, 20] and multiprocessors [18].

Contributions. In this paper, we extend the research on precise scheduling of MC tasks to another dimension. In contrast to prior work that focused on varying the speed of all processors simultaneously, we investigate an alternative approach where the number of active processors may vary in different modes. We formalize the system model and define an MCrp-schedulability problem. To address this problem, we propose two algorithms that are based on virtual deadlines and fluid scheduling, respectively. For each algorithm, we derive a sufficient schedulability test to validate the MCrp-schedulability of the system prior to runtime. To our knowledge, this is the first work on precise scheduling of MC tasks for reserving processors. Furthermore, our schedulability experiments demonstrate the merits of this work over prior related work on varying-speed processors.

Organization. In the rest of this paper, we introduce our system model and problem statement (Sec. II), present two new algorithms based on virtual deadlines (Sec. III) and fluid scheduling (Sec. IV), respectively, as well as their schedulability tests, evaluate the proposed model and algorithms (Sec. V), and conclude (Sec. VI).

II. SYSTEM MODEL AND PROBLEM STATEMENT

We consider the scheduling of a set of $n$ implicit-deadline sporadic MC tasks $\tau = \{\tau_1, \tau_2, \cdots, \tau_n\}$. Each task $\tau_i$ is invoked recurrently with a minimum separation of $\tau$ and conclude (Sec. VI).

Contributions. In this paper, we extend the research on precise scheduling of MC tasks to another dimension. In contrast to prior work that focused on varying the speed of all processors simultaneously, we investigate an alternative approach where the number of active processors may vary in different modes. We formalize the system model and define an MCrp-schedulability problem. To address this problem, we propose two algorithms that are based on virtual deadlines and fluid scheduling, respectively. For each algorithm, we derive a sufficient schedulability test to validate the MCrp-schedulability of the system prior to runtime. To our knowledge, this is the first work on precise scheduling of MC tasks for reserving processors. Furthermore, our schedulability experiments demonstrate the merits of this work over prior related work on varying-speed processors.

Organization. In the rest of this paper, we introduce our system model and problem statement (Sec. II), present two new algorithms based on virtual deadlines (Sec. III) and fluid scheduling (Sec. IV), respectively, as well as their schedulability tests, evaluate the proposed model and algorithms (Sec. V), and conclude (Sec. VI).

II. SYSTEM MODEL AND PROBLEM STATEMENT

We consider the scheduling of a set of $n$ implicit-deadline sporadic MC tasks $\tau = \{\tau_1, \tau_2, \cdots, \tau_n\}$. Each task $\tau_i$ is invoked recurrently with a minimum separation of $\tau_i$ time units, where each invocation is called a job of $\tau_i$ and $\tau_i$ is called the period of $\tau_i$. We also restrict our attention to implicit deadlines. In other words, $T_i$ is also the relative deadline for each task $\tau_i$, and every job of $\tau_i$ has an absolute deadline $T_i$ time units after its release. The worst-case execution time (WCET) of each task $\tau_i$ is estimated a two criticality levels: a low-criticality (L-) estimate $C_i^L$ and a high-criticality (H-) estimate $C_i^H$, where it is assumed that $\forall i, 0 < C_i^L \leq C_i^H \leq T_i$. Furthermore, if $C_i^L = C_i^H$ for task $\tau_i$ so that $\tau_i$ cannot trigger a mode switch as to be described next, then we say $\tau_i$ is a LO-task; by contrast, if $C_i^L < C_i^H$ for task $\tau_i$ so that $\tau_i$ could trigger a mode switch as to be described next, then we say $\tau_i$ is a HI-task. We denote the set of LO-tasks (HI-tasks, respectively) by $\mathcal{T}_{LO}$ ($\mathcal{T}_{HI}$, respectively). We also refer to a job of a LO-task (HI-task, respectively) as LO-job (HI-job, respectively) for short.

Reserving processors and mode switch. We consider a multiprocessor platform consisting of $M^H$ identical processors, each of which has a normalized speed 1.0. In the runtime, if the L-estimates are respected, i.e., all jobs are finished within their L-WCETs, then we say the system is in L-mode; if the L-estimates are exceeded, i.e., some jobs need to execute beyond their L-WCETs and up to their H-WCETs, then we say the system is in H-mode. Note that the H-estimates are assumed to be always respected. In other words, any job that has cumulatively executed for its H-WCET, i.e., $C_i^H$, yet still not completed, is considered as erroneous and would be terminated. That is, only HI-tasks, for which $C_i^L < C_i^H$, could trigger a mode switch. The system begins with L-mode and the amount of execution completed for each job is being monitored during runtime. If any job has cumulatively executed for its L-WCET, i.e., $C_i^L$, but still requires further execution, then the system is immediately notified and switched to H-mode. The system can recover to L-mode once all processors become idle. We require that only $M^L < M^H$ processors are used to actively execute tasks in $\tau$ in L-mode, while the remaining $M^\Delta = (M^H - M^L)$ processors are reserved. Nonetheless, once the system is switched to H-mode, all $M^H$ processors are devoted to execute tasks in $\tau$.

Note that, in contrast to the majority of existing works on MC scheduling, no task is entirely or partially dropped upon a mode switch, and every job must meet its absolute deadline in any system mode. The difference between the two WCET estimates upon mode switch, i.e., $C_i^H - C_i^L$, is compensated by the additional $M^\Delta$ active processors.

In this paper, we assume that the preemption and migration overheads, e.g., due to memory interference, are negligible. Or, equivalently, we assume these overheads are pessimistically taken into account in the WCET estimates.

We denote the utilization of a task $\tau_i$ in L- and H-modes, respectively, by $u_i^L = \frac{C_i^L}{T_i}$ and $u_i^H = \frac{C_i^H}{T_i}$.

Since $C_i^L = C_i^H$ holds for every LO-task, it also holds $u_i^L = u_i^H$ for such task. We further denote the total utilization of the set of LO-tasks and the set of HI-tasks in L- and H-modes, respectively, by

$$U_{LO} = \sum_{\tau_i \in \mathcal{T}_{LO}} u_i^L, \quad U_{HI} = \sum_{\tau_i \in \mathcal{T}_{HI}} u_i^H.$$

We also define $\bar{u}_{HI} = \max_{\tau_i \in \mathcal{T}_{HI}} \{u_i^H\}$ and $\bar{u}_{HI}^L = \max_{\tau_i \in \mathcal{T}_{HI}} \{u_i^L\}$.

Problem Statement. We address the problem of scheduling the MC tasks on $M^H$ unit-speed processors to meet all deadlines in all scenarios with the potential of reserving $M^\Delta$ processors, where $M^\Delta = M^H - M^L > 0$. We say the system is MCrp-schedulable if all deadlines are guaranteed to be met and the following constraints are respected.

- Tasks in $\tau$ only execute on $M^L$ processors if all jobs finish within $C_i^L$ time units;
- Tasks in $\tau$ may execute on all the $M^H$ processors if a any job (of a HI-task) executes for more than $C_i^L$ time units (yet finishes within $C_i^H$ time units of execution).
III. SCHEDULING BY VIRTUAL DEADLINES

In this section, we present a new algorithm to address the problem considered in this paper by leveraging an existing non-MC scheduler fpEDF and the MC scheduling technique of setting virtual deadlines.

A. Algorithm fpEDF and EDF-VD.

For scheduling ordinary non-MC periodic tasks with implicit deadlines, Baruah [1] developed fpEDF (“fp” stands for fixed-priority), which is also applicable to non-MC sporadic tasks with implicit deadline. Under fpEDF, high-utilization tasks for which the utilization exceeds 0.5 are statically prioritized, and the remaining tasks are scheduled according to local earliest-deadline-first (EDF). Assuming $m$ identical unit-speed processors, a utilization-based sufficient schedulability test for fpEDF is as follows.

**Theorem 1** (adapted from Theorem 4 in [1]). Let $U$ denote the total utilization of an implicit-deadline sporadic task set, in which the utilization of task $\tau_i$ is denoted by $u_i$. This task set is schedulable by fpEDF on $m$ identical unit-speed processors if

$$\forall i, u_i \leq 1.0 \quad \text{and} \quad U \leq \frac{m+1}{2}.$$  

On the other hand, EDF-VD (“VD” stands for virtual deadlines) was proposed to address the MC scheduling problem on a single processor and all LO-tasks are dropped upon the mode switch [2, 3]. Under EDF-VD, virtual deadlines are set to promote the execution of HI-tasks in L-mode and to leave slack for the potential extra workload at a mode switch from L to H. A common scaling factor (a constant less than 1.0) is used to determine the virtual deadlines for all HI-tasks. For LO-critical tasks, their virtual deadlines are set identical to their actual deadlines. Then, tasks are scheduled by EDF according to their virtual deadlines in the L-mode and HI-tasks are scheduled by EDF according to their actual deadlines once the system is switched to the H-mode.

B. Algorithm fpEDF-VD-rp

Combining fpEDF and EDF-VD, we propose algorithm fpEDF-VD-rp (“rp” stands for reserving processors) to address the new precise MC scheduling problem in this paper. fpEDF-VD-rp has two phases: a pre-processing phase and a runtime phase.

In the pre-processing phase, a scaling factor $x \in (0, 1)$ is calculated by Eq. (1). The relative virtual deadline of each HI-task $\tau_i$ is set as $x \cdot T_i$, i.e., every HI-job has a virtual deadline $x \cdot T_i$ time units after its release. Furthermore, the number of processors dedicated to LO-tasks $m_{LO}$ is calculated by Eq. (2).

$$x = \max \left\{ u_i^L, \frac{2U_{HI}^L}{M^L - m_{LO} + 1} \right\}$$  

$$m_{LO} = \left\{ \begin{array}{l} \left\lceil \frac{U_{LO}}{2} \right\rceil, \quad \text{if} \quad U_{LO} \leq 1 \vspace{1mm} \\
\left\lceil 2U_{LO} - 1 \right\rceil, \quad \text{if} \quad U_{LO} > 1 \end{array} \right. \quad (1)$$

If $m_{LO} < M^L$ and

$$\max \left\{ u_i^H, \frac{2U_{HI}^H}{M^H - m_{LO} + 1} \right\} + \max \left\{ u_i^H, \frac{2U_{HI}^H}{M^H - m_{LO} + 1} \right\} \leq 1, \quad (3)$$

then the pre-processing phase returns SUCCESS and enter the runtime phase; otherwise, it returns FAILURE. Please note that, by Eq. (1) and Eq. (3), it is clear that $0 < x < 1$ when fpEDF-VD-rp returns SUCCESS.

In the runtime phase, LO-tasks are scheduled in a mode-oblivious manner. The set of all LO-tasks is scheduled on $m_{LO}$ dedicated processors by fpEDF regardless of the mode. Please note that fpEDF reduces to regular uniprocessor EDF when applied on a single processor [1]. By contrast, in L-mode, HI-tasks are scheduled as a set of tasks $\bigcup_{\tau_i \in T_{HI}} \{(x \cdot T_i, C_i^L)\}$ being scheduled by fpEDF on $m_{HI}^L$ dedicated processors, where

$$m_{HI}^L = M^L - m_{LO};$$

upon a mode switch to H-mode, HI-tasks are re-scheduled as a set of tasks $\bigcup_{\tau_i \in T_{HI}} \{(1-x) \cdot T_i, C_i^H\}$ being scheduled by fpEDF on $m_{HI}^H$ dedicated processors, where

$$m_{HI}^H = M^H - m_{LO}.$$  

Note that we use a pair $(T, C)$ to denote an ordinary (non-MC) implicit-deadline sporadic task with period $T$ and WCET $C$.

C. Schedulability Test

We next derive and prove the schedulability test in detail.

**Lemma 1.** All deadlines of LO-tasks are met under fpEDF-VD-rp scheduling if $m_{LO} < M^L < M^H$.

*Proof.* Under fpEDF-VD-rp scheduling, LO-tasks are scheduled by fpEDF on $m_{LO}$ dedicated processors and are not impacted by mode switch. By Eq. (2), it is clear that $U_{LO} \leq (m_{LO}+1)/2$. Furthermore, $\forall i, u_i \leq 1$ must hold in any feasible system. Therefore, by Thm. 1, the lemma follows, given that $m_{LO} < M^L < M^H$ ensures that indeed $m_{LO}$ processors can devote to LO-tasks while leaving some processors for HI-tasks.

**Lemma 2.** All virtual deadlines of HI-tasks are met in L-mode under fpEDF-VD-rp scheduling, if $m_{LO} < M^L < M^H$ and

$$x \geq \max \left\{ u_i^L, \frac{2U_{HI}^L}{M^L - m_{LO} + 1} \right\}$$

*Proof.* In L-mode, HI-tasks are scheduled as a set of ordinary sporadic tasks $\bigcup_{\tau_i \in T_{HI}} \{(x \cdot T_i, C_i^L)\}$ by fpEDF on $m_{HI}^L$ processors. Therefore, by Thm. 1, the deadlines of these ordinary sporadic tasks, i.e., the virtual deadlines of HI-tasks, are met if

$$\sum_{\tau_i \in T_{HI}} \frac{C_i^L}{x \cdot T_i} \leq \frac{m_{HI}^L + 1}{2} \quad \Leftrightarrow \quad \frac{U_{HI}^L}{x} \leq \frac{M^L - m_{LO} + 1}{2} \quad \Leftrightarrow \quad x \geq \frac{2U_{HI}^L}{M^L - m_{LO} + 1}.$$
and $\forall \tau_i \in T_{hi}$, $\frac{C_i^H}{x \cdot T_i} \leq 1$ $\Leftrightarrow$ $\hat{u}_{hi}^H \leq 1$
$\Leftrightarrow$ $x \geq \hat{u}_{hi}^L$.

Thus, the lemma follows.

Lemma 3. All actual deadlines of $hi$-tasks are met in H-mode under fpEDF-VD-rp scheduling, if $m_{lo}^H < M_i^L < M_i^H$ and

$$0 < x \leq 1 - \max \left\{ \frac{2U_{hi}^H}{M^H - m_{lo}^L + 1} \right\}$$

Proof. In H-mode, $hi$-tasks are scheduled as a set of ordinary sporadic tasks $\bigcup_{\tau_i \in T_{hi}} \left\{ \left( (1 - x) \cdot T_i, C_i^H \right) \right\}$ by fpEDF on $m_{hi}^H$ processors. Therefore, by Thm. 1, the actual deadlines of $hi$-tasks are met if

$$\sum_{\tau_i \in T_{hi}} \frac{C_i^H}{(1 - x) \cdot T_i} \leq m_{hi}^H \Leftrightarrow \frac{U_{hi}^H}{1 - x} \leq \frac{M^H - m_{lo}^L + 1}{2}$$
$\Leftrightarrow$ $1 - x \geq \frac{2U_{hi}^H}{M^H - m_{lo}^L + 1}$
$\Leftrightarrow$ $x \leq 1 - \frac{2U_{hi}^H}{M^H - m_{lo}^L + 1}$,

and $\forall \tau_i \in T_{hi}$, $\frac{C_i^H}{(1 - x) \cdot T_i} \leq 1$ $\Leftrightarrow$ $\hat{u}_{hi}^H \leq 1$
$\Leftrightarrow$ $x \leq 1 - \hat{u}_{hi}^H$.

Thus, the lemma follows.

Theorem 2. The system is MCrp-schedulable by fpEDF-VD-rp if $m_{lo}^H < M_i^L$ and Eq. (3) holds.

Proof. This theorem follows directly from the above three lemmas, noting that $x$ is defined by Eq. (1) which implies $x > 0$ and then Eq. (3) implies $x < 1$. That is, meeting virtual deadlines implies meeting their corresponding actual deadlines as well. Please also note that $m_{lo}$ can be easily calculated by Eq. (2) for a given task system.

IV. FLUID SCHEDULING

In this section, we focus on an alternative approach, called the dual-rate fluid scheduling, where each task $\tau_i$ is assigned a constant execution rate in each mode, denoted by $\theta_i^L$ and $\theta_i^H$ in L- and H-modes, respectively. Under fluid scheduling, all tasks conceptually progress simultaneously by “fractions” of a processor at their constant executing rates (per mode, in our particular context). Such simultaneous progression can be implemented by slicing the timeline to smaller pieces or by certain fairness based scheduling algorithms (e.g., DP-Fair [16]), which has been adapted to implement fluid scheduling for MC tasks [15].

A system-wide parameter $\lambda$ and per-task parameters $\theta_i$ for each $hi$-task $\tau_i$ are calculated by:

$$\lambda = \max \left\{ \frac{U_{hi}^L}{M^H - U_{lo} - U_{hi}^H + U_{hi}^L}, \max_{\tau_i \in T_{hi}} \left\{ \frac{u_{hi}^L}{1 + u_{hi}^L - u_{hi}^H} \right\} \right\};$$

$$\forall \tau_i \in T_{hi}, \quad \theta_i = \frac{u_{hi}^L}{\lambda} + u_{hi}^H - u_{hi}^L.$$  

If

$$\lambda \leq \frac{M_i^L - U_{lo} - U_{hi}^H}{U_{hi}^L - U_{hi}^H},$$

then each $hi$-task $\tau_i$ is assigned fluid execution rate $\theta_i^L = \lambda \cdot \theta_i$ in L-mode and a fluid execution rate $\theta_i^H = \theta_i$ in H-mode, whereas each LO-task $\tau_k$ is assigned a fluid execution rate $\theta_k^L = \theta_k^H = u_k^L$ in both modes, and return SUCCESS;

Else return FAILURE.

Noting that $U_{lo} + U_{hi}^H \leq M_i^H$ and $\forall i, u_{hi}^H \leq 1$ must hold for any feasible system, it is clear that Eq. (4) implies $0 < \lambda \leq 1$. 

B. Schedulability Test

We first show that the fluid execution rates assigned by MCF-FR-rp, if they can indeed be satisfied by the underlying platform, are sufficient to ensure all deadline to be met.

**Lemma 4.** If the fluid execution rates assigned by MCF-FR-rp are feasible, then all deadlines must be met.

*Proof. Since \( \theta^L_i = \theta^H_i = u^L_i \) is assigned for each LO-task \( \tau_i \), it is clear that all deadline of LO-tasks are met.

For a HI-task \( \tau_i \), by Eq. (5), \( \theta^L_i = \lambda \theta_i \geq u^L_i \) because \( u^L_i \geq u^L_i \) and \( \theta^H_i = \theta_i \geq u^H_i \) because \( 0 < \lambda \leq 1 \). Therefore, HI-jobs executing entirely in either L- or H-mode must meet their deadlines.

Thus, in the rest of the proof, we only need to focus on the HI-jobs that are released before the mode switch and with a deadline after the mode switch. We consider such a job \( J \) of \( \tau_i \) and let \( t \) denote its release time. If mode switch happens after time \( t + \frac{C^L_i}{\theta^L_i} \), then \( J \) must have finished by time \( t + \frac{C^L_i}{\theta^H_i} \) and therefore meets its deadline; otherwise, \( J \) should have triggered the mode switch at time \( t + \frac{C^L_i}{\theta^H_i} \). On the other hand, because \( \theta^L_i \leq \theta^H_i \), the later the mode switch, the later \( J \) completes its execution, when \( J \) needs to execute for more than \( C^L_i \) (and up to \( C^H_i \)). Therefore, the worst case for \( J \) is when the mode switch happens exactly at time \( t + \frac{C^L_i}{\theta^L_i} \). In this case, \( J \) still must finish by

\[
\begin{align*}
    t + \frac{C^L_i}{\theta^L_i} + \frac{C^H_i - C^L_i}{\theta^H_i} &= t + \frac{C^L_i}{\lambda \theta_i} + \frac{C^H_i - C^L_i}{\theta_i} \\
    &= t + \left( \frac{C^L_i}{\lambda} + C^H_i - C^L_i \right) \frac{1}{\theta_i} \\
    &= t + \frac{C^L_i}{\theta_i} \cdot \frac{T_i}{\theta_i} = t + T_i,
\end{align*}
\]

which is the deadline of \( J \). So, this completes the proof and the lemma follows.

We next show that the fluid execution rates assigned by MCF-FR-rp are indeed feasible.

**Lemma 5.** It holds that \( \forall i, \theta^L_i \leq 1 \land \theta^H_i \leq 1 \).

*Proof.* This is trivially true for LO-tasks. For a HI-task \( \tau_i \), by Eq. (4) we have \( \lambda \geq \frac{u^L_i}{1 + u^L_i - u^L_i} \), and therefore by Eq. (5) we have

\[
\theta^H_i \leq \frac{u^L_i}{1 + u^L_i - u^L_i} + u^H_i - u^L_i = 1.
\]

Furthermore, due to \( 0 < \lambda \leq 1 \), we then have \( \theta^L_i = \lambda \theta_i = \lambda \theta^H_i \leq 1 \), and the lemma follows.

**Lemma 6.** It holds that \( \sum \theta^H_i \leq M^H \).

*Proof.* By Eq. (4), we have \( \lambda \geq \frac{v_{hi}^L}{M^H - U_{hi}^L - U_{hi}^H} \).

Therefore,

\[
\begin{align*}
\sum \theta^H_i &= \sum \theta^H_i + \sum \theta^H_i \\
&= \sum \left( \frac{u^L_i}{\lambda} + u^H_i - u^L_i \right) + \sum \theta_i \\
&= \frac{U_{hi}^L}{U_{hi}^H} + U_{hi}^H - U_{hi}^L + U_{lo} \\
&\leq \frac{U_{hi}^L}{U_{hi}^H} + \frac{M^H - U_{lo} - U_{hi}^L}{U_{hi}^H} + U_{lo} \\
&= M^H.
\end{align*}
\]

The lemma follows.

**Lemma 7.** If Eq. (6) is true, then \( \sum \theta^L_i \leq M^L \) holds.

*Proof.* We have

\[
\begin{align*}
\sum \theta^L_i &= \sum \theta^L_i + \sum \theta^L_i \\
&= \sum \left( \frac{u^L_i}{\lambda} + (u^H_i - u^L_i) \lambda \right) + \sum \theta_i \\
&= \sum \left( u^L_i + (u^H_i - u^L_i) \lambda \right) + \sum \theta_i \\
&= M^L.
\end{align*}
\]

Thus, the lemma follows.

**Theorem 3.** The system is MCrp-schedulable by MCF-FR-rp if Eq. (6) is true where \( \lambda \) is defined by Eq. (4).

*Proof.* Lem. 5, Lem. 6, and Lem. 7 together imply that the fluid execution rates assigned by MCF-FR-rp are feasible if Eq. (6) holds. Therefore, by Lem. 4, this theorem follows.

V. Evaluation

In this section, we conduct schedulability experiments to compare this work to prior work [18] on precise MC scheduling for varying-speed multiprocessors. In [18], all \( m \) processors are active in both L- and H-modes. However, the \( m \) processors must operate at a degraded speed \( \rho < 1.0 \) in L-mode while may run at the full-speed 1.0 in H-mode. In our experiments, we maintain the setting of \( M^H = m \) and \( M^L/M^H = \rho \) so that the total computing capacity of the platform, no matter in L- or H-mode, is the same for the two variants of precise MC scheduling. Then, we compare the schedulability ratios under fpEDF-VD-rp and MCF-FR-rp in this work to that under fpEDF-VD-rp and MCF-FR-rp from [18].

\( ^2\text{We add the “vs” suffix to the algorithms in [18] to emphasize that they are for the varying-speed model and to distinguish them from the algorithms in this paper.} \)
Workload generation. We generate an MC task set by first generating the H-mode utilization of every task, using UUniFast-Discard [9] for given H-mode system utilization. For each task set, we mandate the first task to be a Hi-task to avoid the scenario that all tasks are LO-tasks (which would be in fact non-MC and the techniques discussed in this paper should not be applied). For subsequent tasks, each of them is set to be a Hi-task with probability $P$ or a LO-task with probability $(1 - P)$. If task $\tau_i$ is a Hi-task, then its L-mode utilization $u_{L}^{i}$ is randomly chosen from $[0.2 \times u_{H}^{i}, 0.8 \times u_{H}^{i}]$. By contrast, $u_{P}^{i}$ must equal to $u_{H}^{i}$ for any LO-task $\tau_i$.

Results. In Fig. 2, we report the schedulability results, where $M_{H} = m = 16$, $P = 0.75$, 40 tasks per task set, and 1,000 task sets generated per given H-mode system utilization. Moreover, sub-figures (a), (b), and (c) present the results for which both $M_{L}/M_{H}$ and $\rho$ are 0.25, 0.5, 0.75, respectively. As seen in the figure, both fpEDF-VD-rp and MCF-FR-rp outperform their varying-speed counterpart significantly. In terms of the total number of schedulable task sets (graphically, the “area” between the plot and the x-axis), fpEDF-VD-rp is 1.36 times of its varying-speed counterpart and MCF-FR-rp is 1.5 times of its varying-speed counterpart.

VI. CONCLUSION

In this work, we investigated the precise scheduling of MC tasks for reserving processors in L-mode. We presented two algorithms, called fpEDF-VD-rp and MCF-FR-rp, for this new scheduling problem and provided a sufficient schedulability test for each. Our schedulability experiments demonstrated the effectiveness of the proposed algorithms and the benefits of the new scheduling model.

REFERENCES