12-2014

MC-Fluid: Fluid Model-Based Mixed-Criticality Scheduling on Multiprocessors

Jaewoo Lee
University of Pennsylvania, jaewoo@cis.upenn.edu

Kieu-My Phan

Xiaozhe Gu

Jiyeon Lee

Arvind Easwaran

See next page for additional authors

Recommended Citation

Jaewoo Lee, Kieu-My Phan, Xiaozhe Gu, Jiyeon Lee, Arvind Easwaran, Insik Shin, and Insup Lee, "MC-Fluid: Fluid Model-Based Mixed-Criticality Scheduling on Multiprocessors", *IEEE Real-Time Systems Symposium (RTSS 2014)* , 41-52. December 2014.
http://dx.doi.org/10.1109/RTSS.2014.32

IEEE Real-Time Systems Symposium (RTSS 2014), Rome, Italy, December 2-4, 2014.

MC-Fluid: Fluid Model-Based Mixed-Criticality Scheduling on Multiprocessors

Abstract
A mixed-criticality system consists of multiple components with different criticalities. While mixed-criticality scheduling has been extensively studied for the uniprocessor case, the problem of efficient scheduling for the multiprocessor case has largely remained open. We design a fluid model-based multiprocessor mixed-criticality scheduling algorithm, called *MC-Fluid* in which each task is executed in proportion to its criticality-dependent rate. We propose an exact schedulability condition for MC-Fluid and an optimal assignment algorithm for criticality-dependent execution rates with polynomial-time complexity. Since MC-Fluid cannot be implemented directly on real hardware platforms, we propose another scheduling algorithm, called MC-DP-Fair, which can be implemented while preserving the same schedulability properties as MC-Fluid. We show that MC-Fluid has a speedup factor of $(1+\sqrt{5})/2$ (~ 1.618), which is best known in multiprocessor MC scheduling, and simulation results show that MC-DP-Fair outperforms all existing algorithms.

Disciplines
Computer Engineering | Computer Sciences

Comments
IEEE Real-Time Systems Symposium (RTSS 2014), Rome, Italy, December 2-4, 2014.

Author(s)
Jaewoo Lee, Kieu-My Phan, Xiaozhe Gu, Jiyeon Lee, Arvind Easwaran, Insik Shin, and Insup Lee

MC-Fluid: Fluid Model-based Mixed-Criticality Scheduling on Multiprocessors

Jaewoo Lee*† Kieu-My Phan†‡ Xiaozhe Gu§ Jiyeon Lee‡ Arvind Easwaran§ Insik Shin‡ Insup Lee†
Dept. of Computer and Information Science, University of Pennsylvania, USA†
Dept. of Computer Science, KAIST, South Korea‡
School of Computer Engineering, Nanyang Technological University, Singapore§
E-mail: jaewoo@cis.upenn.edu, insik.shin@cs.kaist.ac.kr, lee@cis.upenn.edu

***Abstract*—A mixed-criticality system consists of multiple components with different criticalities. While mixed-criticality scheduling has been extensively studied for the uniprocessor case, the problem of efficient scheduling for the multiprocessor case has largely remained open. We design a fluid model-based multiprocessor mixed-criticality scheduling algorithm, called *MC-Fluid*, in which each task is executed in proportion to its criticality-dependent rate. We propose an exact schedulability condition for MC-Fluid and an optimal assignment algorithm for criticality-dependent execution rates with polynomial complexity. Since MC-Fluid cannot construct a schedule on real hardware platforms due to the fluid assumption, we propose MC-DP-Fair algorithm, which can generate a non-fluid schedule while preserving the same schedulability properties as MC-Fluid. We show that MC-Fluid has a speedup factor of $(1+\sqrt{5})/2$ (≈ 1.618), which is best known in multiprocessor MC scheduling, and simulation results show that MC-DP-Fair outperforms all existing algorithms.**

I. Introduction

Safety-critical real-time systems such as avionics and automotive are becoming increasingly complex. Recently, there has been a growing attention towards Mixed-Criticality (MC) systems in the real-time community. These systems integrate multiple components with different criticalities onto a single shared platform. Integrated Modular Avionics (IMA) [21] and AUTOSAR [2] are good examples of such systems in industry. These systems consist of low-critical and high-critical components.

A different degree of criticality requires a different level of assurance. The correctness of the high-critical components should be demonstrated under extremely rigorous and pessimistic assumptions. This generally causes large worst-case execution time (WCET) estimates for high-critical components, and such large WCETs could lead to inefficient system designs. While certification authorities (CAs) are concerned with the temporal correctness of only high-critical components, the system designer needs to consider the timing requirement of the entire system under less conservative assumptions. A challenge in MC scheduling is then to simultaneously (1) guarantee the temporal correctness of high-critical components under very pessimistic assumptions, and (2) support the timing requirements of all components, including low-critical ones, under less pessimistic assumptions.

While MC scheduling has been extensively studied for the uniprocessor case, the multiprocessor case has received little attention. In non-MC multiprocessor scheduling, many optimal scheduling algorithms [6], [12], [16] are based on the fluid scheduling model [17], where each task executes in proportion to a static rate (i.e., task utilization). While its proportional progress is still applicable on MC systems, a single static rate is inefficient because characteristics of MC systems change over time. If we apply the worst-case reservation approach, in which tasks are assigned execution rates based on their given criticality-levels, a resulting rate assignment is inefficient because it does not consider dynamics of the MC systems. Using criticality-dependent execution rates, we can find an efficient fluid scheduling algorithm for MC systems.

In this paper, we propose a fluid scheduling algorithm which can compute execution rate of each task depending on a system criticality-level. As the system criticality-level changes at run time, each task will be executed with its criticality-dependent execution rate. A central challenge that we address in this paper is how to determine criticality-dependent execution rates of all the tasks, given that the time instance when the system criticality-level changes is unknown. Even though we have no clairvoyance on the change of the system criticality-level, we can optimally allocate criticality-dependent execution rates to each task within the problem domain.

Contribution. Our contributions are summarized as follows:

- We present a fluid model-based multiprocessor MC scheduling algorithm, called *MC-Fluid*, with criticality-dependent execution rates for each task (Sec. III) and analyze its exact schedulability (Sec. IV). To our best knowledge, this is the first work to apply the fluid scheduling model into MC domain.
- We present an optimal assignment algorithm of execution rates with polynomial complexity (Sec. V).
- We derive the speedup factor of MC-Fluid, $(1+\sqrt{5})/2$, best known in multiprocessor MC scheduling (Sec. VI).
- We propose MC-DP-Fair scheduling algorithm, which can generate a schedule for non-fluid platforms with the same schedulability properties as MC-Fluid (Sec. VII).
- Simulation results show that MC-DP-Fair significantly outperforms the existing approaches (Sec. VIII).

Related Work. Since Vestal [23] introduced a mixed-criticality scheduling algorithm for fixed-priority assignment, the mixed-criticality scheduling problem has received growing attention, in particular, on uniprocessor [3], [4], [7], [8], [14], [15], [23]. For finite jobs, Baruah et al. [7] introduced a priority assignment algorithm, called OCBP (Own Criticality-Based Priority), on dual-criticality systems (low- and high-criticality). It is further extended with sporadic task systems by Baruah et al. [4]. For dynamic-priority assignment, EDF-VD [8] is introduced as an EDF-based scheduling algorithm by which high-criticality tasks are assigned Virtual Deadlines (VDs) with a

*The author was a visiting student at KAIST during the course of this work.

single system-wide parameter. Baruah et al. [3] improved the VD assignment scheme and derived its speedup factor, $4/3$, which is optimal in uniprocessor MC scheduling. Ekberg and Yi [15] presented a VD assignment scheme with task-level parameters and Easwaran [14] improved schedulability with another VD assignment scheme with task-level parameters. With the use of VD, the schedulability result of low-criticality mode can be incorporated into the schedulability analysis of high-criticality mode through the notion of carry-over jobs (a job of a task that is executed across different criticality mode) [3], [14], [15].

Unlike the uniprocessor case, the multiprocessor case has not been much studied [1], [5], [19], [20]. Anderson et al. [1] first considered multiprocessor MC scheduling with a two-level hierarchical scheduler. Pathan [20] proposed a global fixed priority multiprocessor scheduling algorithm for MC task systems. Li et al. [19] introduced a global scheduling algorithm with a speedup factor of $\sqrt{5}+1$. Baruah et al. [5] presented a partitioned scheduling algorithm with a speedup factor of $8/3$.

II. Preliminaries

We study the Mixed-Criticality (MC) scheduling problem on a hard real-time system with m identical processors. In this paper, we consider *dual-criticality* systems with two distinct criticality levels: HI (high) and LO (low).

Tasks. Each MC task is either a LO-criticality task (LO-task) or a HI-criticality task (HI-task). Each MC task τ_i is characterized by $(T_i, C_i^L, C_i^H, \chi_i)$, where $T_i \in \mathbb{R}^+$ is minimum inter-job separation time, $C_i^L \in \mathbb{R}^+$ is LO-criticality WCET (LO-WCET), $C_i^H \in \mathbb{R}^+$ is HI-criticality WCET (HI-WCET), and $\chi_i \in \{HI, LO\}$ is task criticality level. Since HI-WCETs are based on conservative assumptions, we assume that $0 < C_i^L \leq C_i^H \leq T_i$. A task τ_i has a relative deadline equal to T_i. Any task can be executed on at most one processor at any time instant.

Task Sets. We consider a MC sporadic task set $\tau = \{\tau_i\}$, where a task τ_i represents a potentially infinite job release sequence. LO-task set (τ_L) and HI-task set (τ_H) are defined as $\tau_L \stackrel{\text{def}}{=} \{\tau_i \in \tau | \chi_i = LO\}$ and $\tau_H \stackrel{\text{def}}{=} \{\tau_i \in \tau | \chi_i = HI\}$.

Utilization. LO- and HI-task utilization of a task τ_i are defined as $u_i^L \stackrel{\text{def}}{=} C_i^L/T_i$ and $u_i^H \stackrel{\text{def}}{=} C_i^H/T_i$, respectively. System-level utilizations of a task set τ are defined as $U_L^L \stackrel{\text{def}}{=} \sum_{\tau_i \in \tau_L} u_i^L$, $U_H^L \stackrel{\text{def}}{=} \sum_{\tau_i \in \tau_H} u_i^L$, and $U_H^H \stackrel{\text{def}}{=} \sum_{\tau_i \in \tau_H} u_i^H$.

System Modes. The *system mode* is a system-wide variable representing the system criticality level (LO or HI). In LO-mode (the system mode is LO), we assume that no job executes for more than its LO-WCET. In HI-mode, we assume that no job executes for more than its HI-WCET.

MC-schedulability. MC-schedulability indicates both LO- and HI-schedulability: LO-schedulability implies that jobs of all LO- and HI-tasks can complete to execute for their LO-WCETs before their deadlines in LO-mode; and HI-schedulability implies that jobs of all HI-tasks can complete to execute for their HI-WCETs before their deadlines in HI-mode.

The MC System Scenario. We assume the following scenario:

- The system starts in LO-mode. In LO-mode, jobs of all LO- and HI-tasks are released.
- If a job of any HI-task $\tau_i \in \tau_H$ executes for more than its LO-WCET (C_i^L), the system switches the system mode from LO to HI (called *mode-switch*). At mode-switch, the system immediately discards all the jobs of LO-tasks.
- After mode-switch, only the jobs of HI-tasks are released.

If a job of any LO-task $\tau_i \in \tau_L$ (likewise HI-task $\tau_i \in \tau_H$) executes for more than C_i^L in LO-mode (likewise C_i^H in HI-mode), we regard that the system has a fault and do not consider the case. Therefore, we assume that $C_i^L = C_i^H$ for each LO-task $\tau_i \in \tau_L$ without loss of generality.

The problem to determine the time instant of switch-back from HI-mode to LO-mode is beyond the scope of this paper because it is irrelevant to schedulability of MC systems. We can apply the switch-back procedure in Baruah et al. [3].

III. MC-Fluid Scheduling Framework

A. The Fluid Scheduling Platform

Consider a platform where each processing core can be allocated to one or more jobs simultaneously. Each job can be regraded as executing on a dedicated fractional processor with a speed smaller than or equal to one. This scheduling platform is referred to *fluid scheduling platform* [17].

Definition 1 (Fluid scheduling platform [17])**.** *The fluid scheduling platform is a scheduling platform where a job of a task is executed on a fractional processor at all time instants.*

Definition 2 (Execution rate)**.** *A task τ_i is said to be executed with **execution rate** $\theta_i(t_1, t_2) \in \mathbb{R}^+$, s.t. $0 < \theta_i(t_1, t_2) \leq 1$, if every job of the task is executed on a fractional processor with a speed of $\theta_i(t_1, t_2)$ over a time interval $[t_1, t_2]$, where t_1 and t_2 are time instants s.t. $t_1 \leq t_2$*[1].

Schedulability of a fluid platform requires two conditions; (i) task-schedulability (each task has an execution rate that ensures to meet its deadline) and (ii) rate-feasibility (active jobs[2] of all tasks can be executed with their execution rates).

In non-MC multiprocessor systems, many optimal scheduling algorithms [6], [12], [16] have been proposed based on the fluid platform. These algorithms employ a single static rate for each job of a task $\tau_i \in \tau$ from its release to its deadline: $\forall k,\ \theta_i(r_i^k, d_i^k) = \theta_i$ where r_i^k and d_i^k are the release time and deadline of a job J_i^k (the k-th job of task τ_i), respectively. They satisfy task-schedulability by assigning C_i/T_i to θ_i, which is the task utilization of a non-MC task $\tau_i = (T_i, C_i)$ where C_i is its WCET. They satisfy rate-feasibility if $\sum_{\tau_i \in \tau} \theta_i \leq m$.

Lemma 1 presents rate-feasibility condition for fluid model, which can be reused for MC systems. Task-schedulability for MC systems is discussed in Sec. IV.

Lemma 1 (Rate-feasibility, from [6])**.** *Given a task set τ, all tasks can be executed with execution rates iff $\sum_{\tau_i \in \tau} \theta_i \leq m$.*

B. MC-Fluid Scheduling Algorithm

The fluid scheduling algorithm with a single static execution rate per task is inefficient in MC systems[3]. If we assign $\theta_i := u_i^H$ for each HI task $\tau_i \in \tau_H$ and $\theta_i := u_i^L$ for each LO task $\tau_i \in \tau_L$ by the worst-case reservation approach, the result of rate assignment can become overly pessimistic because

[1] Since the task cannot be executed on more than one processor, $\theta_i \leq 1$.

[2] An active job of a task is a job of the task that is released but not finished.

[3] In non-MC multiprocessor systems with an adaptive task model, Block et al. [10] identified inefficiency of a single static rate. To improve soft real-time performance, they adjust the execution rate of a task to exploit spare processing cycles.

task characteristic of MC system can change substantially at mode-switch. According to typical dual-criticality system behaviors, the system changes task characteristic at mode-switch, from executing all LO- and HI-tasks to executing only HI-tasks. Thus, if a scheduling algorithm is allowed to adjust the execution rate of tasks at mode-switch, it can reduce the pessimism of the single rate assignment, considering the dynamics of MC systems. We propose a fluid scheduling algorithm, called *MC-Fluid*, that executes a task with two static execution rates, one for LO-mode and the other for HI-mode.

Definition 3 (MC-Fluid scheduling algorithm)**.** *MC-Fluid is defined with LO- and HI-execution rate (θ_i^L and θ_i^H) for each task $\tau_i \in \tau$. For a job J_i^k of a task τ_i, MC-Fluid assigns θ_i^L to $\theta_i(r_i^k, \min(t_M, d_i^k))$ and θ_i^H to $\theta_i(\max(t_M, r_i^k), d_i^k)$ where r_i^k is its release time, d_i^k is its deadline, and t_M is the time instant of mode-switch. Since all LO-tasks are dropped at mode-switch, θ_i^H is not specified for all LO-tasks $\forall \tau_i \in \tau_L$.*

Informally, MC-Fluid executes each task $\tau_i \in \tau$ with θ_i^L in LO-mode and with θ_i^H in HI-mode. Based on Def. 3, processor resources consuming by a job within a time interval is defined as execution amount.

Definition 4. ***Execution amounts*** *of a job of a task $\tau_i \in \tau$ in a time interval of length t in LO- and HI-mode, denoted by $E_i^L(t)$ and $E_i^H(t)$, are the total amount of processor resources that the job has consumed during this time interval in LO- and HI-mode, respectively: $E_i^L(t) \stackrel{\text{def}}{=} \theta_i^L \cdot t$ and $E_i^H(t) \stackrel{\text{def}}{=} \theta_i^H \cdot t$.*

IV. Schedulability Analysis

In this section, we analyze schedulability of MC-Fluid. MC-schedulability (Theorem 1) consists of LO- and HI-schedulability, which are task-schedulability in LO- and HI-mode (Eqs. (1) and (2)) and rate-feasibility in LO- and HI-mode (Eqs. (3) and (4)).

Theorem 1 (MC-schedulability)**.** *A task set τ, where each task $\tau_i \in \tau$ has LO- and HI-execution rates (θ_i^L and θ_i^H), is MC-schedulable under MC-Fluid iff*

$$\forall \tau_i \in \tau,\ \theta_i^L \geq u_i^L, \tag{1}$$

$$\forall \tau_i \in \tau_H,\ \frac{u_i^L}{\theta_i^L} + \frac{u_i^H - u_i^L}{\theta_i^H} \leq 1, \tag{2}$$

$$\sum_{\tau_i \in \tau} \theta_i^L \leq m, \tag{3}$$

$$\sum_{\tau_i \in \tau_H} \theta_i^H \leq m. \tag{4}$$

To prove Theorem 1, we need to derive task-schedulability condition in LO- and HI-mode because we already know rate-feasibility condition (Lemma 1).

Task-schedulability in LO-mode. Task-schedulability in LO-mode depends only on LO-task utilization.

Lemma 2 (Task-schedulability in LO-mode)**.** *A task $\tau_i \in \tau$ can meet its deadline in LO-mode iff $\theta_i^L \geq u_i^L$.*

Proof: ($\Leftarrow$) Consider a job of the task which is finished in LO-mode. We need to show that the execution amount of the job from its release time (time 0) to its deadline (time T_i) is greater than or equal to LO-WCET (C_i^L). From $\theta_i^L \geq u_i^L$,

$$\theta_i^L \cdot T_i \geq u_i^L \cdot T_i \ \Rightarrow\ E_i^L(T_i) \geq C_i^L. \qquad \text{(by Def. 4)}$$

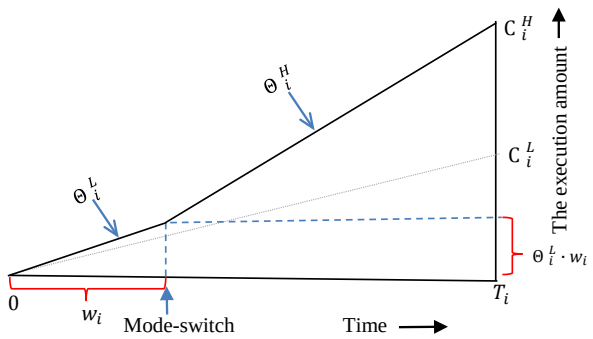


Fig. 1. The model of a carry-over job of a task $\tau_i \in \tau_H$ where mode-switch happens at w_i and the job is executed with θ_i^L and θ_i^H in LO- and HI-mode, respectively (the execution amount of the job until its deadline (T_i) should be C_i^H to meet its deadline).

($\Rightarrow$) We will prove the contrapositive: if $\theta_i^L < u_i^L$, then the task cannot meet its deadline in LO-mode. It is true because $E_i^L(T_i) = \theta_i^L \cdot T_i < u_i^L \cdot T_i = C_i^L$. ■

Task-schedulability in HI-mode. In HI-mode, we do not need to consider task-schedulability of a LO-task because it is dropped at mode-switch. In addition, although a job of a HI-task can be finished in either LO- or HI-mode, we do not need to consider the job finished in LO-mode for task-schedulability in HI-mode.

Consider a job of a HI task τ_i that is finished in HI-mode. The job is released in either LO- or HI-mode. The job in the first case is called *carry-over job*, where the job is released in LO-mode and finished in HI-mode as shown in Fig. 1. Let $w_i \in \mathbb{R}^+$ be the length of a time interval from the release time of the job to mode-switch (or an executed time of the job in LO-mode). We do not need to consider the second case because it is a special situation of the first case when $w_i = 0$, which means that the job is released at mode-switch.

To derive task-schedulability for a carry-over job of τ_i, we need to know the relative time to mode-switch (w_i). We first derive task-schedulability condition with a given w_i. Since mode-switch triggers in the middle of its execution, the job is executed with θ_i^L before mode-switch and with θ_i^H after mode-switch. A cumulative execution amount of the job from its release time to its deadline (T_i) consists of its execution amount from its release time to mode switch with θ_i^L and its execution amount from mode switch to its deadline with θ_i^H:

$$E_i^L(w_i) + E_i^H(T_i - w_i) = \theta_i^L \cdot w_i + \theta_i^H \cdot (T_i - w_i)$$

by Def. 4. Task-schedulability condition with w_i is that the cumulative execution amount of the job is greater than or equal to its HI-WCET (C_i^H):

$$\theta_i^L \cdot w_i + \theta_i^H \cdot (T_i - w_i) \geq C_i^H. \tag{5}$$

Although the value of w_i is required to derive task-schedulability, MC system model assumes that time instant of mode-switch is unknown until runtime scheduling. Thus, we should consider all possible mode-switch scenarios (any valid w_i). Note that $0 \leq w_i \leq T_i$ because mode-switch can happen any time instant between release time and deadline of the job. Then, task-schedulability is Eq. (5) for $\forall w_i$ in $[0, T_i]$:

$$\forall w_i,\ \theta_i^L \cdot w_i + \theta_i^H \cdot (T_i - w_i) \geq C_i^H. \tag{6}$$

To sum up, task-schedulability in HI-mode is Eq. (6).

For concise presentation, we want to eliminate the term of w_i in task-schedulability condition. Its derivation is different depending on whether $\theta_i^L > \theta_i^H$ or $\theta_i^L \leq \theta_i^H$. Lemma 3 considers the first case ($\theta_i^L > \theta_i^H$).

Lemma 3. *A HI-task τ_i with $\theta_i^L > \theta_i^H$ can meet its deadline in HI-mode iff it meets its deadline in HI-mode when $\theta_i^L = \theta_i^H$.*

Proof: ($\Leftarrow$) Let $\theta_i^{L'}$ be the value of the original θ_i^L where $\theta_i^{L'} > \theta_i^H$. Since the task can meet its deadline in HI-mode when $\theta_i^L = \theta_i^H$, from Eq. (6), we have

$$\forall w_i,\ \theta_i^H \cdot w_i + \theta_i^H \cdot (T_i - w_i) = \theta_i^H \cdot T_i \geq C_i^H. \quad (7)$$

To show that the task can meet its deadline in HI-mode when $\theta_i^L = \theta_i^{L'}$, we need to show that Eq. (6) holds: $\forall w_i$,

$$\begin{aligned}\theta_i^{L'} \cdot w_i + \theta_i^H \cdot (T_i - w_i) &> \theta_i^H \cdot w_i + \theta_i^H \cdot (T_i - w_i) \\ &= \theta_i^H \cdot T_i,\end{aligned}$$

which is greater than or equal to C_i^H by Eq. (7).

($\Rightarrow$) Suppose that the task can meet its deadline in HI-mode. Then, we claim that $\theta_i^H \geq u_i^H$. We prove it by contradiction: suppose that $\theta_i^H < u_i^H$. Then, Eq. (6) does not hold when $w_i = 0$:

$$\theta_i^L \cdot 0 + \theta_i^H \cdot (T_i - 0) = \theta_i^H \cdot T_i,$$

which is smaller than C_i^H because $\theta_i^H \cdot T_i < u_i^H \cdot T_i = C_i^H$. However, since we assume that the task meets its deadline in HI-mode, Eq. (6) holds, which is a contradiction. Thus, we proved the claim ($\theta_i^H \geq u_i^H$).

Next, we need to show that the task can meet its deadline in HI-mode when $\theta_i^L = \theta_i^H$. Then, it is required to show that Eq. (6) holds:

$$\forall w_i,\ \theta_i^H \cdot w_i + \theta_i^H \cdot (T_i - w_i) = \theta_i^H \cdot T_i,$$

which is greater than or equal to C_i^H because $\theta_i^H \geq u_i^H$. ∎

Using below corollary, we assume that $\theta_i^L \leq \theta_i^H$ for any task $\tau_i \in \tau_H$ in the rest of the paper.

Corollary 4. *For task-schedulability of a HI-task τ_i in HI-mode, we only need to consider the case where $\theta_i^L \leq \theta_i^H$.*

Proof: Task-schedulability of the task in HI-mode when $\theta_i^L > \theta_i^H$ is equivalent to the one when $\theta_i^L := \theta_i^H$ by Lemma 3. Thus, its task-schedulability in HI-mode is equivalent to the one when $\theta_i^L \leq \theta_i^H$. ∎

Lemma 5 derives task-schedulability in HI-mode by using the assumption that task-schedulability in LO-mode holds. It is a valid assumption because we eventually consider task-schedulability in both LO- and HI-mode for MC-schedulability.

Lemma 5 (Task-schedulability in HI-mode)**.** *Given a HI-task τ_i satisfying task-schedulability in LO-mode, the task can meet its deadline in HI-mode iff*

$$u_i^L/\theta_i^L + (u_i^H - u_i^L)/\theta_i^H \leq 1.$$

Proof: Consider a carry-over job of the task. We first derive the range of a valid w_i and derive task-schedulability in HI-mode by using the range.

Consider the range of w_i, which is $[0, T_i]$. We can further reduce the range by using task-schedulability in LO-mode. The execution amount of the job in LO-mode from its release time to any time instant cannot exceed its LO-WCET (C_i^L)[4]. Thus, its execution amount from its release time to mode-switch also cannot exceed C_i^L:

$$E_i^L(w_i) \leq C_i^L \ \Rightarrow\ \theta_i^L \cdot w_i \leq C_i^L. \quad (8)$$

Combining $0 \leq w_i \leq T_i$ and Eq. (8), we have $0 \leq w_i \leq \min(C_i^L/\theta_i^L, T_i)$. Since task-schedulability in LO-mode holds, we have $\theta_i^L \geq u_i^L$ by Lemma 2. Then,

$$\theta_i^L \geq C_i^L/T_i \ \Rightarrow\ T_i \geq C_i^L/\theta_i^L. \quad \text{(multiplying by } T_i/\theta_i^L\text{)}$$

Thus, the range of valid w_i is $0 \leq w_i \leq C_i^L/\theta_i^L$.

We know that task-schedulability in HI-mode is Eq. (6), which is rewritten to:

$$\begin{aligned}
&\forall w_i,\ \theta_i^L \cdot w_i + \theta_i^H \cdot (T_i - w_i) \geq C_i^H \\
\Leftrightarrow\ &\forall w_i,\ \theta_i^H \cdot T_i \geq (\theta_i^H - \theta_i^L) \cdot w_i + C_i^H \\
\Leftrightarrow\ &\theta_i^H \cdot T_i \geq (\theta_i^H - \theta_i^L) \cdot C_i^L/\theta_i^L + C_i^H \qquad (\because \theta_i^H - \theta_i^L \geq 0)^5 \\
\Leftrightarrow\ &\theta_i^H \cdot T_i \geq C_i^L \cdot \theta_i^H/\theta_i^L + C_i^H - C_i^L \\
\Leftrightarrow\ &1 \geq \frac{u_i^L}{\theta_i^L} + \frac{u_i^H - u_i^L}{\theta_i^H}. \qquad \text{(dividing by } \theta_i^H \cdot T_i\text{)}
\end{aligned}$$

Thus, the task can meet its deadline in HI-mode iff Eq. (6) holds iff $1 \geq u_i^L/\theta_i^L + (u_i^H - u_i^L)/\theta_i^H$. ∎

MC-schedulability. Now, we can prove Theorem 1 by using Lemmas 2 and 5 (task-schedulability in LO- and HI-mode).

Proof of Theorem 1: ($\Leftarrow$) We need to show that the task set is both LO- and HI-schedulable.

From Eq. (1), task-schedulability holds in LO-mode by Lemma 2. From Eq. (3), rate-feasibility holds in LO-mode by Lemma 1. Then, the task set is LO-schedulable.

Since Eq. (2) and task-schedulability in LO-mode hold, task-schedulability in HI-mode holds by Lemma 5. From Eq. (4), rate-feasibility holds in HI-mode by Lemma 1. Then, the task set is HI-schedulable.

($\Rightarrow$) We will prove the contrapositive: if any of the conditions does not hold, then the task set is not MC-schedulable.

If Eq. (1) does not hold, task-schedulability in LO-mode does not hold by Lemma 2. If Eq. (3) does not hold, rate-feasibility in LO-mode does not hold by Lemma 1.

If Eq. (2) does not hold, task-schedulability in HI-mode does not hold by Lemma 5. If Eq. (4) does not hold, rate-feasibility in HI-mode does not hold by Lemma 1. ∎

According to Theorem 1, the worst-case situation is the one where all jobs of HI-tasks are executed for their C_i^L at mode-switch. On fluid platforms, this situation happens because all tasks are always executed with their execution rates whenever they are ready.

	T_i	C_i^L	C_i^H	χ_i	u_i^L	u_i^H	θ_i^L	θ_i^H
τ_1	10	3	8	HI	0.3	0.8	6/10	1
τ_2	20	8	14	HI	0.4	0.7	6/10	9/10
τ_3	30	3	3	HI	0.1	0.1	1/10	1/10
τ_4	40	20	20	LO	0.5	0.5	5/10	

TABLE I
EXAMPLE TASK SET AND ITS EXECUTION RATE ASSIGNMENT.

Example 1. *Consider a two-processor system where its task set τ and its execution rate assignment is given as shown in Table IV. To show that it is schedulable, we need to show that Eqs.* (1)*,* (2)*,* (3) *and* (4) *hold by Theorem 1. We can*

[4]The job triggers mode-switch if the job execute for more than C_i^L.

[5]Note that we already assumed that $\theta_i^L \leq \theta_i^H$ by Corollary 4.

easily check that Eq. (1) *holds. We show that Eq.* (2) *holds:* $\frac{0.3}{6/10}+\frac{0.8-0.3}{1} \leq 1$ *for* τ_1, $\frac{0.4}{6/10}+\frac{0.7-0.4}{9/10} \leq 1$ *for* τ_2, *and* $\frac{0.1}{1/10}+0 \leq 1$ *for* τ_3. *We can check that* $\sum_{\tau_i \in \tau} \theta_i^L = \frac{6+6+1+5}{10} \leq 2$ *for Eq.* (3) *and* $\sum_{\tau_i \in \tau_H} \theta_i^H = \frac{10+9+1}{10} \leq 2$ *for Eq.* (4).

We conclude that the task set is MC-schedulable with the execution rate assignment.

V. The Execution Rate Assignment

We first define the *optimality* of an execution rate assignment algorithm, in a similar way to Davis et al. [13].

Definition 5. *A task set* τ *is called* ***MC-Fluid-feasible*** *if there exists an execution rate assignment that the task set is schedulable under MC-Fluid with. An execution rate assignment algorithm* $\mathbb{A}$ *is called* ***optimal*** *if* $\mathbb{A}$ *can find a schedulable assignment for all MC-Fluid-feasible task sets. For brevity, we refer to an execution rate assignment as an "assignment" and say that a task is feasible when the task is MC-Fluid-feasible.*

In this section, we construct an efficient and optimal assignment algorithm. A naive optimal algorithm checking all combinations of execution rates is intractable because possible real-number execution rates are infinite. Sec. V-A analyzes conditions of an optimal assignment algorithm and formulates it as an optimization problem. Sec. V-B presents a polynomial-complexity algorithm to solve the optimization problem.

A. Conditions for an Optimal Assignment Algorithm

Lemma 6 and Theorem 2 present conditions for an optimal assignment algorithm of θ_i^L and θ_i^H, respectively.

Lemma 6 (An optimal assignment of θ_i^L)**.** *If a task set* τ *is feasible, there is a schedulable assignment where (i)* $\theta_i^L := u_i^L$ *for a task* $\tau_i \in \tau_L$ *and (ii)* $\theta_i^L := \frac{u_i^L \cdot \theta_i^H}{\theta_i^H - u_i^H + u_i^L}$ *for* $\tau_i \in \tau_H$.

Proof: Since the task set is feasible, there exists a schedulable assignment (denoted by $\mathbb{A}$) satisfying Eqs. (1), (2), (3), and (4) by Theorem 1.

(i) Consider a task $\tau_i \in \tau_L$ and its LO-execution rate (θ_i^L) in $\mathbb{A}$. We will show that τ is still schedulable after reassignment of θ_i^L. If we reassign θ_i^L, it only affects Eqs. (1) and (3).

Let θ_i^{L*} be the value of θ_i^L in $\mathbb{A}$. Since $\mathbb{A}$ is schedulable, Eq. (1) holds with θ_i^{L*}, which is $\theta_i^{L*} \geq u_i^L$. Suppose that we reassign $\theta_i^L := u_i^L$. Then, Eq. (1) still holds because $\theta_i^L \geq u_i^L$. Eq. (3) still holds because the decreased θ_i^L (from θ_i^{L*} to u_i^L) does not increase the sum of execution rates.

(ii) Consider a task $\tau_i \in \tau_H$ and its LO-execution rate (θ_i^L) in $\mathbb{A}$. We will show that τ is still schedulable after reassignment of θ_i^L. If we reassign θ_i^L, it only affects Eqs. (1), (2), and (3).

Let θ_i^{L*} and θ_i^{H*} be the value of θ_i^L and θ_i^H in $\mathbb{A}$, respectively. Since Eq. (2) holds for θ_i^{L*} and θ_i^{H*}, we have

$$\frac{u_i^H - u_i^L}{\theta_i^{H*}} \leq 1 - \frac{u_i^L}{\theta_i^{L*}} \;\Rightarrow\; \frac{u_i^H - u_i^L}{\theta_i^{H*}} < 1, \tag{9}$$

since $u_i^L / \theta_i^{L*} > 0$. Eq. (2) with θ_i^{L*} and θ_i^{H*} is rewritten to:

$$\begin{aligned}
&\frac{u_i^L}{\theta_i^{L*}} \leq \frac{\theta_i^{H*} - u_i^H + u_i^L}{\theta_i^{H*}} \\
\Leftrightarrow\; & u_i^L \cdot \theta_i^{H*} \leq \theta_i^{L*}(\theta_i^{H*} - u_i^H + u_i^L) \quad \text{(multiplying by } \theta_i^{H*} \cdot \theta_i^{L*}\text{)} \\
\Leftrightarrow\; & \frac{u_i^L \cdot \theta_i^{H*}}{\theta_i^{H*} - u_i^H + u_i^L} \leq \theta_i^{L*}. \qquad (\because \theta_i^{H*} > u_i^H - u_i^L \text{ by Eq. (9)})
\end{aligned}$$

Since $\mathbb{A}$ is schedulable, Eqs. (1), (2), and (3) hold with θ_i^{L*}. Suppose that we reassign $\theta_i^L := \frac{u_i^L \cdot \theta_i^{H*}}{\theta_i^{H*} - u_i^H + u_i^L}$. Then, Eq. (1) still holds because $\theta_i^L \geq u_i^L$. Eq. (2) still holds because the value of the reassigned θ_i^L is the minimum value satisfying Eq. (2). Eq. (3) still holds because the decreased θ_i^L does not increase the sum of execution rates. ■

Theorem 2 (Conditions for an optimal assignment of θ_i^H)**.** *A task set* τ *is feasible iff there exists an assignment of* θ_i^H *for* $\forall \tau_i \in \tau_H$ *satisfying*

$$u_i^H \leq \theta_i^H, \tag{10}$$

$$U_L^L + U_H^L + \sum_{\tau_i \in \tau_H} \frac{u_i^L (u_i^H - u_i^L)}{\theta_i^H - u_i^H + u_i^L} \leq m, \tag{11}$$

$$\sum_{\tau_i \in \tau_H} \theta_i^H \leq m. \tag{12}$$

Proof: ($\Leftarrow$) To show that the task set is feasible, we need to show that there exists a schedulable assignment.

Consider an assignment where $\theta_i^L := u_i^L$ for each task $\tau_i \in \tau_L$, $\theta_i^L := \frac{u_i^L \cdot \theta_i^H}{\theta_i^H - u_i^H + u_i^L}$ for each task $\tau_i \in \tau_H$, and θ_i^H for each task $\tau_i \in \tau_H$ satisfies Eqs. (10), (11), and (12). To show that this assignment is schedulable by Theorem 1, we need to show that it satisfies Eqs. (1), (2), (3), and (4).

We know that θ_i^L for $\forall \tau_i \in \tau$ satisfies Eq. (1) and θ_i^L for $\forall \tau_i \in \tau_H$ satisfies Eq. (2). We can rewrite Eq. (3) to:

$$\begin{aligned}
\sum_{\tau_i \in \tau} \theta_i^L \leq m &\Leftrightarrow \sum_{\tau_i \in \tau_L} \theta_i^L + \sum_{\tau_i \in \tau_H} \theta_i^L \leq m \\
&\Leftrightarrow \sum_{\tau_i \in \tau_L} u_i^L + \sum_{\tau_i \in \tau_H} \frac{u_i^L \cdot \theta_i^H}{\theta_i^H - u_i^H + u_i^L} \leq m \\
&\Leftrightarrow U_L^L + \sum_{\tau_i \in \tau_H} \left(u_i^L + \frac{u_i^L (u_i^H - u_i^L)}{\theta_i^H - u_i^H + u_i^L} \right) \leq m,
\end{aligned}$$

which is Eq. (11). Then, Eq. (3) holds from Eq. (11). We know that Eq. (12) is Eq. (4). Thus, we showed that the assignment satisfies Eqs. (1), (2), (3), and (4).

($\Rightarrow$) Since the task set is feasible, there is a schedulable assignment where $\theta_i^L = u_i^L$ for $\forall \tau_i \in \tau_L$ and $\theta_i^L = \frac{u_i^L \cdot \theta_i^H}{\theta_i^H - u_i^H + u_i^L}$ by Lemma 6. We need to show that θ_i^H for $\forall \tau_i \in \tau_H$ in the assignment satisfies Eqs. (10), (11), and (12).

Consider θ_i^H of a task $\tau_i \in \tau_H$. Since we already assumed that $\theta_i^L \leq \theta_i^H$ by Corollary 4, we can rewrite $\theta_i^L \leq \theta_i^H$ to:

$$\frac{u_i^L \cdot \theta_i^H}{\theta_i^H - u_i^H + u_i^L} \leq \theta_i^H \;\Rightarrow\; u_i^L \leq \theta_i^H - u_i^H + u_i^L,$$

which is Eq. (10).

Since the assignment is schedulable, Eqs. (3) and (4) hold by Theorem 1. We already proved that Eq. (3) is Eq. (11) in Case $\Leftarrow$ and knew that Eq. (4) is Eq. (12).

Thus, we showed that Eqs. (10), (11), and (12) holds. ■

An optimal assignment algorithm can determine θ_i^L by Lemma 6. Although we have Theorem 2, an assignment algorithm cannot easily determine θ_i^H satisfying the conditions in Theorem 2. So, we present Def. 6, which formulates an optimization problem based on Theorem 2. Lemma 7 shows that an assignment algorithm of θ_i^H by Def. 6 is optimal according to Def. 5.

Definition 6 (Assignment Problem)**.** *Given a task set τ, we define a non-negative real number X_i for each task $\tau_i \in \tau_H$ such that $\theta_i^H := u_i^H + X_i$ and X_i is the solution to the following optimization problem,*

$$\begin{aligned} \textit{minimize} \quad & \sum_{\tau_i \in \tau_H} \frac{u_i^L(u_i^H - u_i^L)}{X_i + u_i^L} \\ \textit{subject to} \quad & \sum_{\tau_i \in \tau_H} X_i - m + U_H^H \le 0, && (\mathsf{CON1}) \\ & \forall \tau_i \in \tau_H, \ -X_i \le 0, && (\mathsf{CON2}) \\ & \forall \tau_i \in \tau_H, \ X_i - 1 + u_i^H \le 0. && (\mathsf{CON3}) \end{aligned}$$

Lemma 7. *An assignment algorithm based on Def. 6 is optimal according to Def. 5.*

Proof: We claim that if the assignment by Def. 6 cannot satisfy Eqs. (10), (11), and (12), no assignment can satisfy those equations. Then, τ is not feasible by Theorem 2.

Suppose that we know a solution to the optimization problem in Def. 6. For each task $\tau_i \in \tau_H$, let X_i^* be the value of X_i in the solution. Let OBJ^* be the value of the objective function with X_i^* for $\forall \tau_i \in \tau_H$. By Def. 6, we have an assignment where $\theta_i^H := u_i^H + X_i^*$ to for $\forall \tau_i \in \tau_H$. Then, Eq. (10) holds from $\mathsf{CON2}$ and $\mathsf{CON3}$ and Eq. (12) holds from $\mathsf{CON1}$.

Suppose that Eq. (11) does not hold with the assignment, meaning $\mathsf{OBJ}^* > m - U_L^L - U_H^L$. For any set of X_i satisfying $\mathsf{CON1}$, $\mathsf{CON2}$, and $\mathsf{CON3}$, we have $\sum_{\tau_i \in \tau_H} \frac{u_i^L(u_i^H - u_i^L)}{X_i + u_i^L} \ge \mathsf{OBJ}^*$ by the definition of the optimization problem and thus Eq. (11) does not hold,

$$\sum_{\tau_i \in \tau_H} \frac{u_i^L(u_i^H - u_i^L)}{X_i + u_i^L} \ge \mathsf{OBJ}^* > m - U_L^L - U_H^L.$$

Thus, no assignment can satisfy Eqs. (10), (11), and (12). ■

B. Convex Optimization for the Assignment Problem

In Sec. V-A, we formulated the optimization problem to construct an optimal assignment algorithm. In this section, we will solve the optimization problem using convex optimization.

Any optimization problem can be rewritten in its dual form using *Lagrange multiplier* (Chapter 5, Boyd et al. [11]), a technique to find a solution for a constrained optimization problem. Using Lagrangian multipliers, we can transform any optimization problem with constraints into its dual problem without constraints. In particular, when its objective function and constraints are differentiable and convex, we can apply *Karush-Kuhn-Tucker* (KKT) conditions (Chapter 5.5.3, [11]), which is a set of optimality conditions for an optimization problem with constraints.

Lemma 8 (KKT conditions [11])**.** *Let $\overline{x}$ is a vector of x_i for $i = 1, \cdots, n$ where $n = |\overline{x}|$ and x_i^* be the value of x_i. Consider an optimization problem:*

$$\textit{minimize } f(\overline{x}) \textit{ subject to } g_j(\overline{x}) \le 0 \textit{ for } j = 1, \cdots, N$$

where N is the number of $g_j(\overline{x})$ and all of $f(\overline{x})$ and $g_j(\overline{x})$ for $\forall j$ are differentiable and convex. Then, $\overline{x^*}$ minimizes $f(\overline{x})$ iff there exists $\overline{\lambda^*}$ s.t.

$$\begin{aligned} & \forall j, \ g_j(\overline{x^*}) \le 0, && (13) \\ & \forall j, \ \lambda_j^* \cdot g_j(\overline{x^*}) = 0, && (14) \\ & \forall i, \ \frac{\partial f(\overline{x^*})}{\partial x_i} + \sum_j (\lambda_j^* \frac{\partial g_j(\overline{x^*})}{\partial x_i}) = 0, && (15) \\ & \forall j, \ \lambda_j^* \ge 0, && (16) \end{aligned}$$

where λ_j is a Lagrange multiplier, λ_j^* is the value of λ_j, and $\overline{\lambda}$ is a vector of λ_j.

Lemma 9 applies KKT conditions (Lemma 8) to the optimization problem in Def. 6 because the objective function and all the constraints are differentiable and convex. Then, we only need to find the value of Lagrange multipliers satisfying KKT condition for the optimal solution.

For brevity of this section, we abbreviate $\forall \tau_i \in \tau_H$ as $\forall \tau_i$ because only HI-tasks are considered in Def. 6. We use ψ, λ_i, and ν_i as Lagrange multipliers for $\mathsf{CON1}$, $\mathsf{CON2}$, and $\mathsf{CON3}$, respectively. We denote a vector of X_i, λ_i, and ν_i by $\overline{X}$, $\overline{\lambda}$, and $\overline{\nu}$, and denote the value of X_i, λ_i, and ν_i by X_i^*, λ_i^*, and ν_i^*, respectively. We define, for each task τ_i,

$$\mathsf{Cost}_i(x) \stackrel{\mathsf{def}}{=} \frac{u_i^L(u_i^H - u_i^L)}{(x + u_i^L)^2} \quad \text{where } x \in \mathbb{R}^+.$$

Lemma 9. *Consider the optimization problem in Def. 6. $\overline{X^*}$ minimizes $f(\overline{X})$ iff there exist ψ^*, $\overline{\lambda^*}$, and $\overline{\nu^*}$ s.t.*

$$\begin{aligned} & \sum_{\tau_i} X_i^* - m + U_H^H \le 0, && (17) \\ & \forall \tau_i, \ -X_i^* \le 0, \qquad X_i^* - 1 + u_i^H \le 0, && (18) \\ & \psi^*(\sum_{\tau_i} X_i^* - m + U_H^H) = 0, && (19) \\ & \forall \tau_i, \ \lambda_i^*(-X_i^*) = 0, \qquad \nu_i^*(X_i^* - 1 + u_i^H) = 0, && (20) \\ & \forall \tau_i, \ -\mathsf{Cost}_i(X_i^*) + \psi^* - \lambda_i^* + \nu_i^* = 0, && (21) \\ & \psi^* \ge 0 \wedge \forall \tau_i(\lambda_i^* \ge 0 \wedge \nu_i^* \ge 0), && (22) \end{aligned}$$

where $f(\overline{X}) = \sum_{\tau_i} \frac{u_i^L(u_i^H - u_i^L)}{X_i + u_i^L}$.

Proof: We will derive KKT conditions for the optimization problem in Def. 6: $f(\overline{X}) = \sum_{\tau_i} \frac{u_i^L(u_i^H - u_i^L)}{X_i + u_i^L}$ and $g_1(\overline{X}) = \sum_{\tau_i} X_i - m + U_H^H$ for $\mathsf{CON1}$; for $\forall \tau_i$, $g_{2,i}(\overline{X}) = -X_i$ for $\mathsf{CON2}$ and $g_{3,i}(\overline{X}) = X_i - 1 + u_i^H$ for $\mathsf{CON3}$.

We know that $f(\overline{X})$ and all constraints are differentiable. To show that they are convex, we need to show that their second derivatives are no smaller than zero:

$$\forall \tau_i, \ \forall X_i \ , \frac{\partial^2 f(\overline{X})}{\partial (X_i)^2} = \frac{2 \cdot u_i^L(u_i^H - u_i^L)}{(X_i + u_i^L)^3} \ge 0 \text{ and}$$

$$\forall \tau_i, \ \forall \tau_j, \ \forall X_i, \ \frac{\partial^2 g_1(\overline{X})}{\partial (X_i)^2} = \frac{\partial^2 g_{2,j}(\overline{X})}{\partial (X_i)^2} = \frac{\partial^2 g_{3,j}(\overline{X})}{\partial (X_i)^2} = 0.$$

We derive KKT conditions for the problem by Lemma 8:

- We denote Lagrange multipliers for $g_1(\overline{X})$, $g_{2,i}(\overline{X})$ and $g_{3,i}(\overline{X})$ by ψ, λ_i and ν_i, respectively.
- Eq. (13) for $g_1(\overline{X})$ is Eq. (17). Eq. (13) for $g_{2,i}(\overline{X})$ and $g_{3,i}(\overline{X})$ is Eq. (18).
- Eq. (14) for $g_1(\overline{X})$ is Eq. (19). Eq. (14) for $g_{2,i}(\overline{X})$ and $g_{3,i}(\overline{X})$ is Eq. (20).

- Eq. (15) is $\forall \tau_i,\ \partial f(\overline{X})/\partial X_i + \psi - \lambda_i + \nu_i = 0$, which is Eq. (21).
- Eq. (16) for $g_1(\overline{X})$, $g_{2,i}(\overline{X})$ and $g_{3,i}(\overline{X})$ is Eq. (22).

By Lemma 8, $\overline{X^*}$ minimizes $f(\overline{X})$ iff there exist ψ^*, $\overline{\lambda^*}$, and $\overline{\nu^*}$ satisfying Eqs. (17), (18), (19), (20), (21), and (22). ■

Example 2. *Consider the system in Example 1. We need to solve the optimization problem in Def. 6. Suppose that we have $\overline{X^*}$ satisfying KKT conditions in Lemma 9 for the system: $X_1^* = 0.2$, $X_2^* = 0.2$, and $X_3^* = 0$. Eq.* (17) *holds:* $0.2 + 0.2 - 2 + 1.6 = 0$. *We can easily check that Eqs.* (18) *and* (19) *hold. To satisfy Eq.* (20)*, we have $\lambda_1^* = \lambda_2^* = \nu_2^* = \nu_3^* = 0$. By Eq.* (21)*, we have $-\frac{0.15}{(0.2+0.3)^2} + \psi^* + \nu_1^* = 0$, $-\frac{0.12}{(0.2+0.4)^2} + \psi^* = 0$, and $\psi^* - \lambda_3^* = 0$. Then, we have $\psi^* = 1/3$, $\nu_1^* = 4/15$, and $\lambda_3^* = 1/3$. Eq.* (22) *holds with ψ^*, $\overline{\lambda^*}$, and $\overline{\nu^*}$. Thus, $\overline{X^*}$ is the solution to the problem in Def. 6 by Lemma 9.*

By Def. 6, we have $\theta_1^H := 1$, $\theta_2^H := 0.9$, and $\theta_3^H := 0.1$. We can assign θ_i^L for $\forall \tau_i \in \tau$ by Lemma 6. Since this calculated assignment is the same as the assignment in Example 1, we conclude that the assignment in Example 1 is optimal.

The OERA algorithm. General KKT conditions are not easily solvable because we cannot find the feasible values of the Lagrange multipliers for the conditions. We propose the *Optimal Execution Rate Assignment (OERA) algorithm* to solve our KKT conditions. In our KKT conditions (Lemma 9), we note that λ_i^* and ν_i^* for each task τ_i depend only on X_i^*. Since only ψ^* depends on $\overline{X^*}$, if we know ψ^* satisfying our KKT conditions, we can independently analyze λ_i^*, ν_i^*, and X_i^* for each task τ_i. Based on this observation, we develop a simple greedy algorithm.

Before presenting the OERA algorithm, Lemma 10 shows that this observation is correct.

Lemma 10. *Given a task τ_i and ψ (≥ 0), if we assign $X_i :=$*

$$\mathsf{Cal\_X_i}(\psi) \stackrel{\mathsf{def}}{=} \begin{cases} 0 & \text{if } \psi \geq \mathsf{Cost}_i(0), \\ 1 - u_i^H & \text{if } \psi < \mathsf{Cost}_i(1-u_i^H), \\ \sqrt{\frac{u_i^L(u_i^H - u_i^L)}{\psi}} - u_i^L & \text{otherwise}, \end{cases}$$

then Eqs. (18)*,* (20)*, and* (21) *hold with some λ_i (≥ 0) and ν_i (≥ 0).*

Proof: To satisfy Eq. (18), we assume that $0 \leq X_i \leq 1 - u_i^H$. Then, to prove this lemma, we only need to show that Eqs. (20) and (21) hold in each case of $\mathsf{Cal\_X_i}(\psi)$.

Case ($\psi \geq \mathsf{Cost}_i(0)$)**.** For some $\nu_i \geq 0$, Eq. (21) is simplified to:

$$\psi \leq \mathsf{Cost}_i(X_i) + \lambda_i, \tag{23}$$

by removing ν_i. Note that $\mathsf{Cost}(X_i) \leq \mathsf{Cost}(0)$ where $0 \leq X_i \leq 1 - u_i^H$. To satisfy Eq. (23), λ_i should satisfy that $\lambda_i \geq \psi - \mathsf{Cost}_i(X_i) \geq \psi - \mathsf{Cost}_i(0)$, which is greater than or equal to 0 from the assumption. To satisfy Eq. (20), we assign $\nu_i := 0$ and $X_i := 0$. Thus, Eqs. (20) and (21) hold.

Case ($\psi < \mathsf{Cost}_i(1-u_i^H)$)**.** For some $\lambda_i \geq 0$, Eq. (21) is simplified to:

$$\psi + \nu_i \geq \mathsf{Cost}_i(X_i), \tag{24}$$

by removing λ_i. Note that $\mathsf{Cost}(1 - u_i^H) \leq \mathsf{Cost}(X_i)$ where $0 \leq X_i \leq 1 - u_i^H$. To satisfy Eq. (24), ν_i should satisfy that $\nu_i \geq \mathsf{Cost}_i(X_i) - \psi \geq \mathsf{Cost}_i(1 - u_i^H) - \psi$, which is greater than 0 from the assumption. To satisfy Eq. (20), we assign $\lambda_i := 0$ and $X_i := 1 - u_i^H$. Thus, Eqs. (20) and (21) hold.

Case ($\mathsf{Cost}_i(1 - u_i^H) \leq \psi < \mathsf{Cost}_i(0)$)**.** To satisfy Eq. (20), we assign $\lambda_i := 0$ and $\nu_i := 0$. Then, to satisfy Eq. (21), we need to satisfy $\psi = \mathsf{Cost}_i(X_i)$, from which, X_i can be computed:

$$\psi = \frac{u_i^L(u_i^H - u_i^L)}{(X_i + u_i^L)^2} \ \Leftrightarrow\ X_i = \sqrt{\frac{u_i^L(u_i^H - u_i^L)}{\psi}} - u_i^L.$$

Combining three cases, we showed that Eqs. (18), (20), and (21) hold with some positive λ_i and ν_i. ■

To find ψ satisfying our KKT condition, we divide the range of ψ. According to Lemma 10, if ψ is greater than or equal to $\mathsf{Cost}_i(0)$ or smaller than $\mathsf{Cost}_i(1 - u_i^H)$, $\mathsf{Cal\_X_i}(\psi)$ is a constant; otherwise, $\mathsf{Cal\_X_i}(\psi)$ is a linear function of $1/\sqrt{\psi}$. To divide the range of ψ, we define an ordered set $G \stackrel{\mathsf{def}}{=} \{x \in \mathbb{R}^+ | x = \mathsf{Cost}_i(0)$ or $x = \mathsf{Cost}_i(1 - u_i^H)$ for $\forall \tau_i\}$, sorted in increasing order. We denote the j-th element of G as G_j. In OERA, we utilize the property that $\mathsf{Cal\_X_i}(\psi)$ for some task τ_i changes a constant to a linear function of $1/\sqrt{\psi}$ or vice-versa at $\psi = G_j$.

We present the OERA algorithm (Algorithm 1). Suppose that ψ^* satisfying our KKT conditions is greater than zero. We assign $X_i := \mathsf{Cal\_X_i}(\psi^*)$ for $\forall \tau_i$, which satisfies Eqs. (18), (20), (21), and (22) by Lemma 10. To satisfy the remaining conditions, Eqs. (17) and (19), we require the condition that $\sum_{\tau_i} X_i - m + U_H^H = 0$, which is $\sum_{\tau_i} \mathsf{Cal\_X_i}(\psi^*) = m - U_H^H$.

The LHS of the required condition is a piecewise linear function of $1/\sqrt{\psi}$. To handle the piecewise function, we note that it only changes at $G_j \in G$. By examining the value of the LHS of the condition in each G_j, we can find the range of ψ^* where some ψ^* satisfies the condition (Line 2). If the range is found, the LHS of the condition is just a linear function within the range. Then, we can find ψ^* by solving the condition (Line 3) and OERA returns $\overline{X^*}$ where $X_i^* = \mathsf{Cal\_X_i}(\psi^*)$ (Line 4).

However, it is possible that there does not exist positive ψ^* satisfying our KKT conditions, which means that ψ^* satisfying our KKT conditions is zero because it is non-negative. OERA returns $\overline{X^*}$ where $X_i^* = \mathsf{Cal\_X_i}(0)$ for $\forall \tau_i$ (Line 7). Theorem 3 proves the correctness of OERA.

Algorithm 1 The OERA algorithm

Input: τ_H, G, and m
Output: $\overline{X^*}$ satisfying KKT conditions in Lemma 9
1: **for** $j := 1$ to $|G| - 1$ **do**
2: **if** $\sum_{\tau_i} \mathsf{Cal\_X_i}(G_{j+1}) \leq m - U_H^H < \sum_{\tau_i} \mathsf{Cal\_X_i}(G_j)$ **then**
3: find ψ^* by solving $\sum_{\tau_i} \mathsf{Cal\_X_i}(\psi^*) = m - U_H^H$
4: **return** $\overline{X^*}$ where $X_i = \mathsf{Cal\_X_i}(\psi^*)$
5: **end if**
6: **end for**
7: **return** $\overline{X^*}$ where $X_i = \mathsf{Cal\_X_i}(0)$

Example 3. *Consider KKT conditions in Example 2. We have $G = \{0.245, 0.6, 0.75, 1.667\}$. We apply Algorithm 1. In Fig. 2, X-axis represents $1/\sqrt{\psi}$. Blue (solid) line is $\sum_{\tau_i} \mathsf{Cal\_X_i}(\psi)$, which is a piecewise linear function where its slope is changed at each $1/\sqrt{G_j}$ where $G_j \in G$. Red (dashed) line is a constant function of $m - U_H^H$. When*

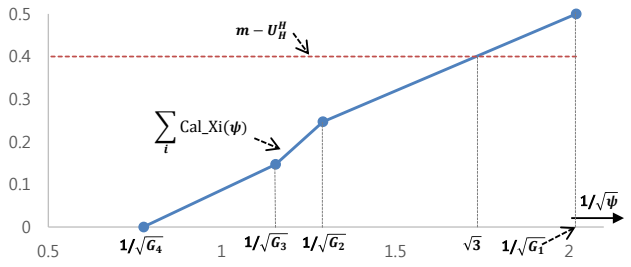


Fig. 2. The plot of $\sum_{\tau_i} \mathsf{Cal\_X_i}(\psi)$ in respect to $1/\sqrt{\psi}$ on Example 3, which is a piecewise linear function (note that there is ψ^* s.t. $\sum_{\tau_i} \mathsf{Cal\_X_i}(\psi^*) = 0.4$ where $G_1 \le \psi^* < G_2$).

$G_1(= 0.245) \le \psi^* < G_2(= 0.6)$, the condition in Line 2 holds: $\sum_{\tau_i} \mathsf{Cal\_X_i}(G_2) \le m - U_H^H < \sum_{\tau_i} \mathsf{Cal\_X_i}(G_1)$, which is $0.5 \le 0.4 < 0.247$. In other words, a crossing point between blue line and red line is located in $(1/\sqrt{G_2}, 1/\sqrt{G_1}]$. Then, we can find ψ^* by solving $\sum_{\tau_i} \mathsf{Cal\_X_i}(\psi^*) = m - U_H^H$, where blue line meets red line at $1/\sqrt{\psi^*}$. Since $0.2 + \sqrt{0.12/\psi^*} - 0.4 = 0.4$, we have $1/\sqrt{\psi^*} = \sqrt{3}$. Finally, $\psi^* = 1/3$, $X_1^* = 0.2$, $X_2^* = \sqrt{\frac{0.12}{1/3}} - 0.4 = 0.2$, and $X_3^* = 0$.

We conclude that OERA can find $\overline{X^*}$ satisfying the KKT condition in Lemma 9 for Example 2.

Theorem 3. *Algorithm 1 (OERA) can find $\overline{X^*}$ satisfying KKT conditions in Lemma 9.*

Proof: (i) [Line 1-6] Suppose that there exists $\psi^*(> 0)$ satisfying our KKT conditions. Consider some $\psi^* > 0$. We assign $X_i := \mathsf{Cal\_X_i}(\psi^*)$ for $\forall \tau_i$. By Lemma 10, Eqs. (18), (20), (21), and (22) hold. To satisfy Eqs. (17) and (19), we require the condition $\sum_{\tau_i} X_i - m + U_H^H = 0$, which is $\sum_{\tau_i} \mathsf{Cal\_X_i}(\psi^*) = m - U_H^H$.

We need to find ψ^* satisfying the required condition. Since the LHS of the condition is a piecewise-linear increasing function of $1/\sqrt{\psi}$, we can find a unique $\psi^* \ge 0$ satisfying the condition if $\sum_{\tau_i} \mathsf{Cal\_X_i}(\psi') \le m - U_H^H$ where ψ' is the maximum value of ψ^* and $\sum_{\tau_i} \mathsf{Cal\_X_i}(\psi'') > m - U_H^H$ where ψ'' is the minimum value of ψ^*. We know that $\sum_{\tau_i} \mathsf{Cal\_X_i}(\max_j G_j) = 0$, which is smaller than or equal to $m - U_H^H$. However, we do not know whether $\sum_{\tau_i} \mathsf{Cal\_X_i}(\min_j G_j) > m - U_H^H$ or not. If it holds, the algorithm can find a unique solution of ψ^* satisfying the required condition and return $\overline{X^*}$ where $X_i^* = \mathsf{Cal\_X_i}(\psi^*)$ satisfying the KKT conditions. If it does not hold ($\sum_{\tau_i} \mathsf{Cal\_X_i}(\min_j G_j) \le m - U_H^H$), there is no solution when $\psi^* > 0$.

(ii) [Line 7] Suppose that there does not exist $\psi^*(>0)$ satisfying the KKT conditions. Then, Line 1-6 cannot find a solution, which means $\sum_{\tau_i} \mathsf{Cal\_X_i}(\min_j G_j) \le m - U_H^H$. Then, we assign $\psi^* := 0$ because ψ^* is non-negative from Eq. (22). We assign $X_i := \mathsf{Cal\_X_i}(0)$ for $\forall \tau_i$. By Lemma 10, Eqs. (18), (20), (21), and (22) hold. To satisfy Eqs. (17) and (19), we require the condition $\sum_{\tau_i} \mathsf{Cal\_X_i}(0) \le m - U_H^H$.

For a task τ_i, we claim that $\mathsf{Cal\_X_i}(0) = \mathsf{Cal\_X_i}(\min_j G_j)$. If $\min_j G_j = 0$, we have $\mathsf{Cal\_X_i}(0) = \mathsf{Cal\_X_i}(\min_j G_j)$. Otherwise, we have $\min_j G_j > 0$. When $\min_j G_j \ne \mathsf{Cost}_i(1 - u_i^H)$, by the second case of Lemma 10, we know that $\mathsf{Cal\_X_i}(0) = \mathsf{Cal\_X_i}(\min_j G_j)$. When $\min_j G_j = \mathsf{Cost}_i(1 - u_i^H)$, we need to apply the third case of Lemma 10. Since $\mathsf{Cal\_X_i}(\psi)$ calculates X_i s.t. $\psi = \mathsf{Cost}(X_i)$ in the third case of Lemma 10, we have $\mathsf{Cal\_X_i}(\mathsf{Cost}_i(1 - u_i^H)) = 1 - u_i^H$, from which we have $\mathsf{Cal\_X_i}(\min_j G_j) = 1 - u_i^H$ by using $\mathsf{Cost}_i(1 - u_i^H) = \min_j G_j$. Since $\mathsf{Cal\_X_i}(0) = 1 - u_i^H$, we have $\mathsf{Cal\_X_i}(0) = \mathsf{Cal\_X_i}(\min_j G_j)$. Thus, we proved the claim.

By the claim above, we have $\sum_{\tau_i} \mathsf{Cal\_X_i}(0) = \sum_{\tau_i} \mathsf{Cal\_X_i}(\min_j G_j)$, which is no greater than $m - U_H^H$ from the assumption. Thus, the required condition holds.

Since the KKT conditions hold with $\psi^* = 0$, the algorithm returns $\overline{X^*}$ where $X_i^* = \mathsf{Cal\_X_i}(\psi^*)$ satisfying the KKT conditions ■

Algorithm Complexity. Let n be $|\tau|$. Algorithm 1 (OERA) can find a solution at most $O(n)$ iteration because $|G| \le 2|\tau_H| \le 2n$. Since each iteration takes $O(n)$ time to solve the equation in Line 3, OERA has polynomial complexity.

VI. The Speedup Factor

In this section, we quantify the effectiveness of MC-Fluid via the metric of processor speedup factor [18]. In general, the speedup factor of an algorithm $\mathbb{A}$ is defined as a real number α (≥ 1) such that any task set schedulable by an optimal clairvoyant algorithm[6] on m speed-*1* processors is also schedulable by $\mathbb{A}$ on m speed-α processors. In other words, a task set that is clairvoyantly schedulable on m speed-$(1/\alpha)$ processors is also schedulable by $\mathbb{A}$ on m speed-*1* processors. We consider this special task set (clairvoyantly schedulable on m speed-$(1/\alpha)$) for speedup factor derivation.

Lemma 11 shows $\alpha \ge 4/3$ for non-clairvoyant algorithms.

Lemma 11. *No non-clairvoyant algorithm for scheduling dual-criticality task set on multiprocessors can have a speedup factor better than $4/3$.*

Proof: It follows from Theorem 5 of Baruah et al. [3] since MC scheduling on uniprocessor is a special case of MC scheduling on m processors. ■

For the special task set for speedup derivation, we present a sufficient schedulability condition with MC-Fluid.

Lemma 12. *Given a task set τ that is clairvoyantly schedulable on m speed-$(1/\alpha)$ processors, the task set is schedulable on m speed-1 processors by MC-Fluid if there is $x \in \mathbb{R}^+$ s.t.*

$$0 \le x \le 1, \tag{25}$$

$$U_L^L + U_H^L + \frac{U_H^H - U_H^L}{(\alpha - 1)x + 1} \le m, \tag{26}$$

$$U_H^H + (\alpha - 1)U_H^L \cdot x \le m. \tag{27}$$

Proof: To show that τ is feasible by MC-Fluid, we need to show that there exists an assignment satisfying Eqs. (10), (11), and (12) by Theorem 2.

Since τ is clairvoyantly schedulable on m speed-$(1/\alpha)$ processors, we assume that $U_L^L + U_H^L \le m/\alpha$ and $U_H^H \le m/\alpha$. Since the task is assumed to be executable on a speed-$(1/\alpha)$ processor, we assume that $u_i^L \le u_i^H \le 1/\alpha$ for $\tau_i \in \tau$.

Let $\mathbb{A}$ be an assignment where $\theta_i^H := u_i^H + (\alpha - 1)u_i^L \cdot x$ for $\forall \tau_i \in \tau_H$ satisfying Eq. (25), (26), and (27). We show that the assigned θ_i^H by x is no greater than 1 from the definition of the execution rate (Def. 2), for $\forall \tau_i \in \tau_H$,

$$\begin{aligned}
\theta_i^H &= u_i^H + (\alpha - 1)u_i^L \cdot x \\
&\le u_i^H + (\alpha - 1)u_i^L && \text{(from Eq. (25))} \\
&\le u_i^H + (\alpha - 1)u_i^H && (\because u_i^L \le u_i^H) \\
&\le 1. && \text{(from } u_i^H \le 1/\alpha)
\end{aligned}$$

[6]In MC systems, a clairvoyant (fluid or non-fluid) scheduling algorithm is the one that knows the time instant of mode-switch before runtime scheduling.

(i) We show that $\mathbb{A}$ satisfies Eq. (10): from Eq. (25), for $\forall \tau_i \in \tau_H$,

$$0 \le (\alpha-1)u_i^L \cdot x \;\Rightarrow\; u_i^H \le u_i^H + (\alpha-1)u_i^L \cdot x \;\Rightarrow\; u_i^H \le \theta_i^H,$$

which is Eq. (10).

(ii) We show that $\mathbb{A}$ satisfies Eq. (11): from Eq. (26),

$$\begin{aligned} & U_L^L + U_H^L + \frac{U_H^H - U_H^L}{(\alpha-1)\cdot x + 1} \le m \\ \Rightarrow\; & U_L^L + U_H^L + \sum_{\tau_i \in \tau_H} \frac{u_i^H - u_i^L}{(\alpha-1)\cdot x + 1} \le m \\ \Rightarrow\; & U_L^L + U_H^L + \sum_{\tau_i \in \tau_H} \frac{u_i^L(u_i^H - u_i^L)}{\theta_i^H - u_i^H + u_i^L} \le m, \\ & \qquad (\because (\alpha-1)u_i^L \cdot x + u_i^L = \theta_i^H - u_i^H + u_i^L) \end{aligned}$$

which is Eq. (11).

(iii) We show that $\mathbb{A}$ satisfies Eq. (12): from Eq. (27),

$$U_H^H + (\alpha-1)U_H^L \cdot x \le m \Rightarrow \sum_{\tau_i \in \tau_H} \left(u_i^H + (\alpha-1)u_i^L \cdot x\right) \le m \Rightarrow \sum_{\tau_i \in \tau_H} \theta_i^H \le m,$$

which is Eq. (12). ∎

Now, we consider the range of feasible x satisfying Eqs. (25), (26) and (27). We will find the lower bound of the feasible x. Eq. (26) can be rewritten to:

$$\begin{aligned} & U_L^L + U_H^L + \frac{U_H^H - U_H^L}{(\alpha-1)\cdot x + 1} \le m \\ \Leftrightarrow\; & \frac{U_H^H - U_H^L}{(\alpha-1)\cdot x + 1} \le m - U_L^L - U_H^L \\ \Leftrightarrow\; & \frac{U_H^H - U_H^L}{m - U_L^L - U_H^L} \le (\alpha-1)\cdot x + 1 \\ & \qquad (\because m - U_L^L - U_H^L > m/\alpha - U_L^L - U_H^L \ge 0) \\ \Leftrightarrow\; & \frac{U_H^H + U_L^L - m}{m - U_L^L - U_H^L} \le (\alpha-1)\cdot x \\ \Leftrightarrow\; & \frac{U_H^H + U_L^L - m}{(\alpha-1)(m - U_L^L - U_H^L)} \le x. \qquad (\because \alpha - 1 > 0) \end{aligned} \tag{28}$$

Since LHS of Eq. (28) is non-negative, the lower bound of x is $\frac{U_H^H + U_L^L - m}{(\alpha-1)(m-U_L^L-U_H^L)}$ by Eqs. (25) and (28).

We will find the upper bound of the feasible x. Eq. (27) can be rewritten to $x \le \frac{m - U_H^H}{(\alpha-1)U_H^L}$. Since $(\alpha-1)U_H^L + U_H^H \le (\alpha-1)U_H^H + U_H^H \le m$, we have

$$(\alpha-1)U_H^L + U_H^H \le m \Rightarrow 1 \le \frac{m - U_H^H}{(\alpha-1)U_H^L}.$$

Thus, the upper bound of x is 1 by Eqs. (25) and (27).

From the range of the feasible x, we can derive the condition for the existence of x satisfying Eqs. (25), (26), and (27):

$$\frac{U_H^H + U_L^L - m}{(\alpha-1)(m - U_L^L - U_H^L)} \le 1. \tag{29}$$

If Eq. (29) holds, MC-Fluid can schedule the special task set for speedup factor derivation by Lemma 12.

We derive the speedup factor of MC-Fluid using Eq. (29).

Theorem 4. *The speedup factor of MC-Fluid is* $(1+\sqrt{5})/2$.

Proof: Consider a task set schedulable by any clairvoyant algorithm on m speed-$(1/\alpha)$ processors. To prove this theorem, we need to show that the task set can be scheduled by MC-Fluid on m speed-*1* processors and $\alpha = (1+\sqrt{5})/2$. Since Eq. (29) is a sufficient schedulability condition of MC-Fluid for the task set, we can derive Eq. (30) as another sufficient schedulability condition of MC-Fluid:

$$\begin{aligned} & U_H^H + U_L^L - m \le (\alpha-1)(m - U_L^L - U_H^L) \\ \Leftarrow\; & U_H^H + (\alpha-1)U_H^L + \alpha \cdot U_L^L \le \alpha \cdot m \\ \Leftarrow\; & m/\alpha + (\alpha-1)(m/\alpha - U_L^L) + \alpha \cdot U_L^L \le \alpha \cdot m \\ & \qquad (\because U_L^L + U_H^L \le m/\alpha \text{ and } U_H^H \le m/\alpha) \\ \Leftarrow\; & m + U_L^L \le \alpha \cdot m \\ \Leftarrow\; & m + m/\alpha \le \alpha \cdot m. \qquad (\because U_L^L \le U_L^L + U_H^L \le m/\alpha) \end{aligned} \tag{30}$$

Eq. (30) holds when $\alpha \ge (1+\sqrt{5})/2$ because Eq. (30) is $\alpha^2 - \alpha - 1 \ge 0$ and α is a positive value.

Since Eq. (30) holds with $\alpha = (1+\sqrt{5})/2$, MC-Fluid can schedule any task set that is clairvoyantly schedulable on m speed-$(1/\alpha)$ processors. Thus, the speedup factor of MC-Fluid is $(1+\sqrt{5})/2$. ∎

Baruah et al. [3] showed that any non-clairvoyant uniprocessor algorithm for a dual-criticality task set cannot have a speedup factor better than $4/3$. While this result is also applicable for multiprocessors, we do not know an exact lower bound on a speedup factor for any non-clairvoyant multiprocessor algorithm yet. In this work, we show that a sufficient lower bound for multiprocessors is $(1+\sqrt{5})/2$.

VII. MC-DP-Fair Scheduling Algorithm

Many fluid-based scheduling algorithms, including MC-Fluid, rely on the fractional (fluid) processor assumption, and this assumption makes them infeasible to construct a schedule on real (non-fluid) hardware platforms. Overcoming the limitation of fluid-based algorithms, several approaches (e.g., [6], [12], [16]) have been introduced to construct a non-fluid schedule for real hardware platforms, while holding an equivalent schedulability to that of a fluid-based schedule. Such approaches differ in the unit of a time interval over which they enforce the equivalence of fluid-based and non-fluid schedules. Quantum-based approaches (e.g., [6]) identify the minimal scheduling unit (i.e., a time quantum) in hardware platforms: every time quantum, they enforce the execution of every task to satisfy that the difference of the execution amount between the actual schedule and the fluid schedule is no greater than 1. Deadline partitioning approaches (e.g., [12], [16]) enforce every task to meet the fluid scheduling requirement only every distinct deadline of the system, which suffices with respect to schedulability.

DP-Fair. We choose DP-Fair [16] due to its simplicity. DP-Fair enforces the fluid requirement every *time slice*, defined as a time interval between two consecutive *Deadline Partitions (DPs)*, where a DP is defined as a distinct release time or deadline from all jobs in the system. For a time slice, DP-Fair ensures that every task gets executed for its required execution amount until the end of the time slice, which satisfies the fluid requirement within the time slice.

The required execution amount for a job of a non-MC task τ_i within a time slice of interval length l is calculated as $l \cdot \delta_i$

where δ_i is density of the task ($\delta_i = C_i/T_i$ where T_i is its period and C_i is its WCET).

We recapitulate the schedulability properties of DP-Fair in the following lemmas.

Lemma 13 (from [16])**.** *Given a non-MC task set τ and a time slice, if the task set is scheduled within the time slice under DP-Fair and $\sum_{i\in\tau}\delta_i \le m$, then the required execution amount of each task within the time slice can be executed until the end of the time slice.*

Lemma 14 (from [16])**.** *A non-MC task set τ is schedulable under DP-Fair iff $\sum_{\tau_i\in\tau}\delta_i \le m$.*

MC-DP-Fair. Building upon DP-Fair, we propose *MC-DP-Fair* scheduling algorithm, which constructs a non-fluid schedule based on MC-Fluid. In MC-DP-Fair, DP-Fair should be extended considering the characteristics of MC-Fluid: 1) LO- and HI-density should be modified to satisfy the fluid execution requirement of MC-Fluid in the end of a time slice; and 2) the worst-case scenarios of MC-Fluid and MC-DP-Fair should be the same.

Definition 7 (MC-DP-Fair scheduling algorithm)**.** *MC-DP-Fair is defined with a per-task virtual deadline and a special DP. For each task $\tau_i \in \tau$, we define a virtual deadline $V_i \in \mathbb{R}^+$ s.t. $0 < V_i \le T_i$. Let Γ denote the earliest DP after time instant of mode-switch. According to DP-Fair, MC-DP-Fair executes each task $\tau_i \in \tau$ with V_i before Γ and with T_i after Γ as its deadline.*

According to Def. 7, the DP Γ is one of virtual deadlines of jobs because deadline partitioning is performed based on virtual deadlines of jobs in the system and Γ is the earliest DP after mode-switch. Note that scheduling policy of MC-DP-Fair is changed not at mode-switch but at Γ, which is the earliest DP after mode-switch. By this delayed scheduling policy switch, MC-DP-Fair has the same worst-case situation as MC-Fluid does.

In LO-mode, MC-DP-Fair considers, for a task $\tau_i \in \tau$, $\delta_i^L \stackrel{\text{def}}{=} C_i^L/V_i$, where V_i is the virtual deadline of τ_i. Then, the amount of execution required for a task τ_i and time slice length l is $l \cdot \delta_i^L$ and any job of the task can be executed for C_i^L until its virtual deadline in LO-mode.

In HI-mode, we can derive the density of task depending on whether the time instant of mode-switch is a DP or not. We claim that we only need to consider the first case. Consider that mode-switch happens in the middle of a time slice. Note that Γ indicates the end of this time slice. MC-DP-Fair executes HI-tasks for the amount of their required remaining execution (calculated based on C_i^L) until Γ. Then, the second case is equivalent to the case where mode-switch happens at Γ, which is a DP.

Now, consider the case where mode-switch happens at a DP. The density of the job depends on the remaining execution amount to C_i^H and the remaining time to deadline (T_i) at the DP. We first calculate the remaining time to deadline of the job: $T_i - w_i$ where w_i is time interval length from its release time to the DP. Next, we calculate the amount of remaining execution up to C_i^H by using Lemma 13. If we assume that $\sum_{\tau_i\in\tau}\delta_i^L \le m$, we can compute the execution amount of the job from its release time to the DP: $E_i^L(w_i) = \delta_i^L \cdot w_i$. We can compute δ_i^H after the DP:

$$\delta_i^H \stackrel{\text{def}}{=} \frac{C_i^H - E_i^L(w_i)}{T_i - w_i} = \frac{C_i^H - \delta_i^L \cdot w_i}{T_i - w_i}. \tag{31}$$

Then, the amount of execution required for the job and time slice length l in HI-mode is $l \cdot \delta_i^H$ and the job is finished its execution in its deadline in HI-mode.

Virtual Deadline Assignment. In this section, we denote the values of θ_i^L and θ_i^H in the optimal assignment of MC-Fluid (by θ_i^{L*} and θ_i^{H*}), respectively. Intuitively, to utilize MC-Fluid analysis, the required LO-density for a HI-task is no greater than θ_i^{L*} and the required LO-density for a LO-task is no greater than its task utilization. Def. 8 proposes a virtual deadline assignment according to the optimal assignment of MC-Fluid. Lemma 15 validates the correctness of Def. 8.

Definition 8 (Virtual deadline assignment for MC-DP-Fair)**.** *We assign $V_i := T_i$ for a LO-task $\tau_i \in \tau_L$ and $V_i := C_i^L/\theta_i^{L*}$ for a HI-task $\tau_i \in \tau_H$ where θ_i^{L*} is the value of θ_i^L in the optimal assignment for MC-Fluid.*

Lemma 15. *Given a feasible MC task set τ, if the task set is scheduled by MC-DP-Fair with the virtual deadline assignment by Def. 8, then (i) $\delta_i^L \le \theta_i^{L*}$ for each task $\tau_i \in \tau$ and (ii) $\delta_i^H \le \theta_i^{H*}$ for each task $\tau_i \in \tau_H$.*

Proof: (i) We show that $\delta_i^L \le \theta_i^{L*}$ for $\tau_i \in \tau$: we have $\delta_i^L = C_i^L/T_i = u_i^L = \theta_i^{L*}$ for $\tau_i \in \tau_L$ by Lemma 6 and $\delta_i^L = C_i^L/V_i = \theta_i^{L*}$ for $\tau_i \in \tau_H$.

(ii) We will show that $\delta_i^H \le \theta_i^{H*}$ for $\tau_i \in \tau_H$. Since δ_i^H varies on w_i in Eq. (31), we will derive the maximum δ_i^H and show that the value is no greater than θ_i^{H*}. Since we assume $\sum_{\tau_i\in\tau}\delta_i^L \le m$ in derivation of Eq. (31), we need to check that the assumption holds: it holds because $\delta_i^L \le \theta_i^{L*}$ from Case (i) and $\sum_{\tau_i\in\tau}\theta_i^{L*} \le m$ from the feasible task set.

To find the maximum δ_i^H, consider derivative of Eq. (31):

$$\frac{d\ \delta_i^H}{d\ w_i} = \frac{-\delta_i^L(T_i - w_i) + (C_i^H - \delta_i^L \cdot w_i)}{(T_i - w_i)^2} = \frac{C_i^H - \delta_i^L \cdot T_i}{(T_i - w_i)^2},$$

which is greater than or equal to 0 if $u_i^H \ge \delta_i^L$.

To show that the derivative is non-negative, we show that $u_i^H \ge \delta_i^L$. Since $\theta_i^{L*} \ge \delta_i^L$ from Case (i), we only need to show $u_i^H \ge \theta_i^{L*}$: since $\theta_i^{L*} = \frac{u_i^L \cdot \theta_i^{H*}}{\theta_i^{H*} - u_i^H + u_i^L}$ by Lemma 6,

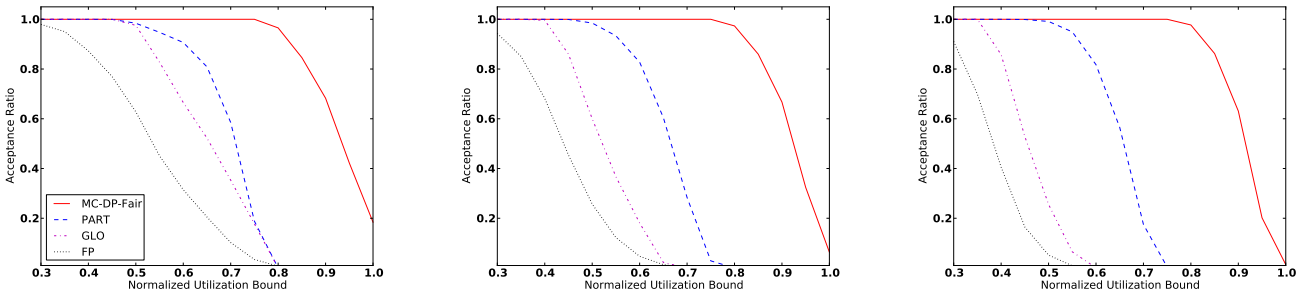
$$u_i^H \ge \frac{u_i^L \cdot \theta_i^{H*}}{\theta_i^{H*} - u_i^H + u_i^L} \Leftrightarrow u_i^H(\theta_i^{H*} - u_i^H + u_i^L) \ge u_i^L \cdot \theta_i^{H*}$$
$$\Leftrightarrow \theta_i^{H*}(u_i^H - u_i^L) \ge u_i^H(u_i^H - u_i^L),$$

which is true because θ_i^{H*} satisfies Eq. (10), which is $\theta_i^{H*} \ge u_i^H$, by Theorem 2.

From the derivative of δ_i^H, we can find the maximum δ_i^H: the maximum δ_i^H can be found at the maximum w_i because $\frac{d\ \delta_i^H}{d\ w_i} \ge 0$ and thus δ_i^H is an increasing function of w_i. Since the task use virtual deadline in LO-mode, we have $w_i \le V_i$. Then, we can calculate δ_i^H when $w_i = V_i$:

$$\delta_i^H = \frac{C_i^H - \delta_i^L \cdot V_i}{T_i - V_i} = \frac{C_i^H - C_i^L/V_i \cdot V_i}{T_i - C_i^L/\theta_i^{L*}}$$
$$= \frac{u_i^H - u_i^L}{1 - u_i^L/\theta_i^{L*}},$$

which is θ_i^{H*} by Lemma 6. Thus, we conclude $\delta_i^H \le \theta_i^{H*}$. ■



(a) m=2 (b) m=4 (c) m=8

Fig. 3. The acceptance ratio with varying the normalized utilization bound (U^b/m) and the number of processors (m).

Schedulability Analysis. Since MC tasks are subject to different execution time requirements (and thereby different densities), we extend Lemma 14 for MC systems as follows.

Lemma 16. *Given δ_i^L and δ_i^H for each task $\tau_i \in \tau$, a MC task set τ is MC-schedulable iff*

$$\sum_{\tau_i \in \tau} \delta_i^L \leq m, \tag{32}$$

$$\sum_{\tau_i \in \tau_H} \delta_i^H \leq m. \tag{33}$$

Proof: Eq. (32) is LO-schedulability by Lemma 14 with LO-mode. Eq. (33) is HI-schedulability by Lemma 14 with HI-mode. Since MC-schedulability implies both LO- and HI-schedulability, Eqs. (32) and (33) are MC-schedulability. ■

Based on Lemmas 15 and 16, Theorem 5 presents that MC-DP-Fair has the same schedulability as MC-Fluid.

Theorem 5. *A MC task set τ is schedulable by MC-DP-Fair with the virtual deadline assignment by Def. 8 iff τ is MC-Fluid-feasible.*

Proof: ($\Rightarrow$) To show that τ is feasible, we need to show that there exists an assignment satisfying Eq. (10), (11), and (12) by Theorem 2. Since θ_i^{H*} for $\forall \tau_i \in \tau_H$ can only violate Eq. (11) according to Def. 6, we need to show that Eq. (11) holds.

Since the task set is schedulable by MC-DP-Fair with Def. 8, we know that Eq. (32) holds by Lemma 16. Eq. (32) is rewritten to:

$$\begin{aligned}
\sum_{\tau_i \in \tau} \delta_i^L \leq m \Leftrightarrow & \sum_{\tau_i \in \tau_L} C_i^L/T_i + \sum_{\tau_i \in \tau_H} C_i^L/V_i \leq m \\
\Leftrightarrow\ & U_L^L + \sum_{\tau_i \in \tau_H} \frac{u_i^L \cdot \theta_i^{H*}}{\theta_i^{H*} - u_i^H + u_i^L} \leq m \\
\Leftrightarrow\ & U_L^L + \sum_{\tau_i \in \tau_H} \left(u_i^L + \frac{u_i^L (u_i^H - u_i^L)}{\theta_i^{H*} - u_i^H + u_i^L} \right) \leq m \\
\Leftrightarrow\ & U_L^L + U_H^L + \sum_{\tau_i \in \tau_H} \frac{u_i^L (u_i^H - u_i^L)}{\theta_i^{H*} - u_i^H + u_i^L} \leq m,
\end{aligned}$$

which is Eq. (11).

($\Leftarrow$) To show that the task set is schedulable, we need to show that Eqs. (32) and (33) hold by Lemma 16.

Since the task set is feasible, Eq. (11) with θ_i^{H*} holds by Theorem 2. We already showed that Eq. (11) is Eq. (32) in Case $\Rightarrow$.

Since the feasible task set is scheduled by MC-DP-Fair with Def. 8, we have $\delta_i^H \leq \theta_i^{H*}$ for each task $\tau_i \in \tau_H$ by Lemma 15. We show that Eq. (33) holds:

$$\sum_{\tau_i \in \tau_H} \delta_i^H \leq \sum_{\tau_i \in \tau_H} \theta_i^{H*} \leq m,$$

which is true because the optimal assignment satisfies CON1 in Def. 6 with $\theta_i^{H*} = X_i + u_i^H$. ■

VIII. Simulation

In this section, we will evaluate the performance of the MC-Fluid framework. We compare the schedulability of MC-DP-Fair (a non-fluid algorithm with the same schedulability as MC-Fluid) with previously published MC-scheduling approaches on multiprocessors: the global fpEDF algorithm (GLO) [19], the partitioned EDF algorithm (PART) [5], and the global fixed-priority algorithm (FP) [20]. The speedup factors of MC-DP-Fair, GLO, and PART are $(1+\sqrt{5})/2$ (≈ 1.618), $1+\sqrt{5}$ (≈ 3.236), and $8/3$ (≈ 2.667), respectively.

Task Set Generation. We generate random task sets according to the workload-generation algorithm [19]. Let U^b be the upper bound of system utilization in both LO- and HI-mode. Input parameters are U^b, m (the number of processors), Z^b (the upper bound of task utilization), and P^c (the probability of task criticality). Initially, $m = 2$, $Z^b = 0.7$, and $P^c = 0.5$. We will also evaluate varying different input parameters. A random task is generated as follows (all task parameters are randomly drawn in uniform distribution):

- u_i^L is a real number drawn from the range [0.02, Z^b].
- T_i is an integer drawn from the range [20, 300].
- R_i (the ratio of u_i^H/u_i^L) is a real number drawn from the range [1, 4].
- P_i (the probability that the task is a HI-task) is a real number from the range [0,1]. If $P_i < P^c$, set $\chi_i := LO$ and $C_i^L := \lfloor u_i^L \cdot T_i \rfloor$. Otherwise, set $\chi_i := HI$, $C_i^L := \lfloor u_i^L \cdot T_i \rfloor$, and $C_i^H := \lfloor u_i^L \cdot R_i \cdot T_i \rfloor$.

Repeat to generate a task in the task set until $\max(U_H^L + U_L^L,\ U_H^H)$ is larger than U^b. Then, discard the task added last.

Simulation Results. Fig. 3 shows the acceptance ratio (ratio of schedulable task sets) over varying $m \in \{2, 4, 8\}$ and normalized utilization bound U^b/m from 0.3 to 1.0 in step of 0.05. Each data point is based on 10,000 task sets. The result shows that MC-DP-Fair outperforms previously known approaches.

Fig. 4 and 5 show the effect of varying different parameters (P^c or Z^b). We use the weighted acceptance ratio [9] to reduce

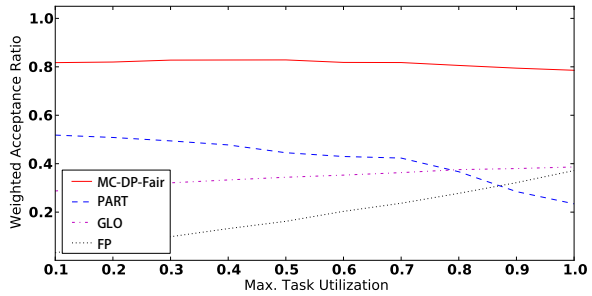


Fig. 4. The weighted acceptance ratio with varying the upper bound of task utilization (Z^b).

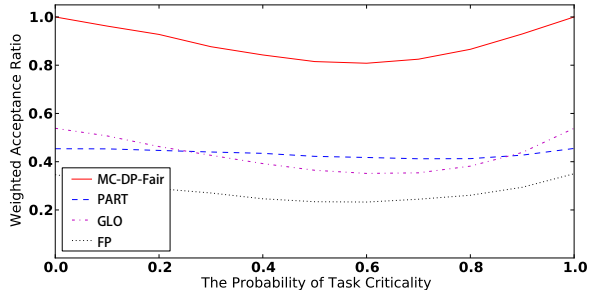


Fig. 5. The weighted acceptance ratio with varying the probability of task criticality (P^c).

the number of dimensions in the plots. Let U_m^b be U^b/m and $A(U_m^b)$ be the acceptance ratio for U_m^b. The weighted acceptance ratio $W(S)$ is calculated to:

$$W(S) \stackrel{\text{def}}{=} \frac{\sum_{U_m^b \in S} \left(U_m^b \cdot A(U_m^b)\right)}{\sum_{U_m^b \in S} U_m^b},$$

where S is the set of U^b/m. The set S is the same as the one for Fig. 3 ($S = \{0.3, 0.35, \cdots, 1.0\}$). In Figs. 4 and 5, each data point is based on 15,000 task sets where 1,000 task sets are experimented for each U^b/m in S.

Fig. 4 shows the weighted acceptance ratio varying the upper bound of task utilization (Z^b). MC-DP-Fair and GLO are insensitive to Z^b while the performance of PART decreases as Z^b increases due to difficulty of scheduling large utilization tasks. On the other hand, the performance of FP increases as Z^b increases because interference-based analysis favors a smaller number of tasks[7].

Fig. 5 shows the weighted acceptance ratio varying the probability of task criticality (P^c). MC-DP-Fair can schedule all task sets when only LO-tasks or only HI-tasks are generated (i.e., $P^c = 0$ or $P^c = 1$) since MC-DP-Fair generalizing DP-Fair is also optimal for the non-MC task model.

IX. Conclusion

We presented a multiprocessor mixed-criticality scheduling algorithm, called MC-Fluid, based on the fluid scheduling platform. Given LO- and HI-execution rates per task, we derived an exact schedulability analysis of MC-Fluid on the dual-criticality systems. We also presented an optimal rate assignment algorithm with polynomial complexity. For standard (non-fluid) platforms, we presented MC-DP-Fair scheduling algorithm, which has the same scheduling properties as MC-Fluid. We showed that MC-Fluid has a speedup factor of $(1+\sqrt{5})/2$ (≈ 1.618), which is best known in multiprocessor MC scheduling, and MC-DP-Fair outperforms all existing algorithms in simulation results.

As future work, we plan to derive a tighter speedup factor and apply another schedule generation algorithm for non-MC platforms (e.g., RUN, which is based on a weak notion of the fluid scheduling model [22]) to reduce preemption overheads, under the MC-Fluid framework. We also plan to improve the MC-Fluid framework itself by considering more than two execution rates for better schedulability.

[7]While FP uses interference-based analysis, all others use utilization-based analysis.

Acknowledgement

This work was supported in part by BSRP (NRF-2010-0006650, NRF-2012R1A1A1014930), NCRC (2012-0000980), IITP (2011-10041313, 14-824-09-013) and KIAT (M002300089) funded by the Korea Government (MEST/MSIP/MOTIE). In addition, it was supported in part by ARO W911NF-11-1-0403, ONR N000141310802, and MoE Tier-2 grant number MOE2013-T2-2-029.

References

[1] J. H. Anderson, S. K. Baruah, and B. B. Brandenburg. Multicore operating-system support for mixed criticality. In *Workshop on Mixed Criticality*, 2009.
[2] AUTOSAR. AUTomotive Open System ARchitecture. www.autosar.org.
[3] S. Baruah, V. Bonifaci, G. D"Angelo, H. Li, A. Marchetti-Spaccamela, S. Van der Ster, and L. Stougie. The preemptive uniprocessor scheduling of mixed-criticality implicit-deadline sporadic task systems. In *ECRTS*, 2012.
[4] S. Baruah, A. Burns, and R. Davis. Response-time analysis for mixed criticality systems. In *RTSS*, 2011.
[5] S. Baruah, B. Chattopadhyay, H. Li, and I. Shin. Mixed-criticality scheduling on multiprocessors. *Real-Time Systems*, 2013.
[6] S. Baruah, N. K. Cohen, C. G. Plaxton, and D. A. Varvel. Proportionate progress: a notion of fairness in resource allocation. In *Symposium on the Theory of Computing (STOC)*, 1993.
[7] S. Baruah, H. Li, and L. Stougie. Towards the design of certifiable mixed-criticality systems. In *RTAS*, 2010.
[8] S. K. Baruah, V. Bonifaci, G. D'Angelo, A. Marchetti-Spaccamela, S. Van Der Ster, and L. Stougie. Mixed-criticality scheduling of sporadic task systems. In *European Symposium on Algorithms (ESA)*, 2011.
[9] A. Bastoni, B. B. Brandenburg, and J. H. Anderson. Cache-related preemption and migration delays: Empirical approximation and impact on schedulability. In *OSPERT workshop*, 2010.
[10] A. Block, J. Anderson, and G. Bishop. Fine-grained task reweighting on multiprocessors. In *RTCSA*, 2005.
[11] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.
[12] H. Cho, B. Ravindran, and E. D. Jensen. An optimal real-time scheduling algorithm for multiprocessors. In *RTSS*, 2006.
[13] R. Davis and A. Burns. Improved priority assignment for global fixed priority pre-emptive scheduling in multiprocessor real-time systems. *Real-Time Systems*, 2011.
[14] A. Easwaran. Demand-based scheduling of mixed-criticality sporadic tasks on one processor. In *RTSS*, 2013.
[15] P. Ekberg and W. Yi. Bounding and shaping the demand of mixed-criticality sporadic tasks. In *ECRTS*, 2012.
[16] S. Funk, G. Levin, C. Sadowski, I. Pye, and S. Brandt. DP-Fair: a unifying theory for optimal hard real-time multiprocessor scheduling. *Real-Time Systems*, 2011.
[17] P. Holman and J. H. Anderson. Adapting pfair scheduling for symmetric multiprocessors. *J. Embedded Comput.*, 2005.
[18] B. Kalyanasundaram and K. Pruhs. Speed is as powerful as clairvoyance. *Journal of the ACM*, 2000.
[19] H. Li and S. Baruah. Global mixed-criticality scheduling on multiprocessors. In *ECRTS*, 2012.
[20] R. Pathan. Schedulability analysis of mixed-criticality systems on multiprocessors. In *ECRTS*, 2012.
[21] P. Prisaznuk. Integrated modular avionics. In *Aerospace and Electronics Conference*, 1992.
[22] P. Regnier, G. Lima, E. Massa, G. Levin, and S. Brandt. RUN: Optimal Multiprocessor Real-Time Scheduling via Reduction to Uniprocessor. In *RTSS*, 2011.
[23] S. Vestal. Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance. In *RTSS*, 2007.