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 \((1 + \sqrt{5})/2 \approx 1.618\), 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
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 $\tau_i$ 
is characterized by $(T_i, C^L_i, C^H_i, \chi_i)$, where $T_i \in \mathbb{R}^+$ 
is minimum inter-job separation time, $C^L_i \in \mathbb{R}^+$ is 
LO-criticality WCET (LO-WCET), $C^H_i \in \mathbb{R}^+$ 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 $0 < C^L_i \leq C^H_i \leq T_i$. A task $\tau_i$ 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 $\tau_i$ represents a potentially infinite job release 
sequence. The LO-task set ($\tau_L$) and HI-task set ($\tau_H$) are defined 
as $\tau_L \equiv \{\tau_i \mid \chi_i = LO\}$ and $\tau_H \equiv \{\tau_i \mid \chi_i = HI\}$.

**Utilization.** LO- and HI-task utilization of a task $\tau_i$ are defined 
as $u^L_i \overset{def}{=} C^L_i/T_i$ and $u^H_i \overset{def}{=} C^H_i/T_i$, respectively.

System-level utilizations of a task set $\tau$ are defined as $U^L_{\tau} \overset{def}{=} \sum_{\tau_i \in \tau_L} u^L_i$, $U^H_{\tau} \overset{def}{=} \sum_{\tau_i \in \tau_H} u^H_i$, and $U_{\tau} \overset{def}{=} \sum_{\tau_i \in \tau} u^H_i$.

**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 ($C^L_i$), 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 $\tau_j \in \tau_L$ (likewise HI-task $\tau_j \in \tau_H$) 
exectues for more than $C^L_j$ in LO-mode (likewise $C^H_j$ in HI-
mode), we regard that the system has a fault and do not 
consider the case. Therefore, we assume that $C^L_j = C^H_j$ for 
each LO-task $\tau_j \in \tau_L$ 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 regarded 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 $\tau_i$ 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. $t_1 \leq t_2$.

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 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: 
$\forall k, \theta_i(r^k_i, d^k_i) = \theta_i$ where $r^k_i$ and $d^k_i$ are the release time and 
deadline of a job $J^k_i$ (the k-th job of task $\tau_i$), respectively. They 
satisfy task-schedulability by assigning $C_i/T_i$ to $\theta_i$, 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 $\tau$, all 
tasks can be executed with execution rates $\theta_i$ if $\sum_{\tau_i \in \tau} \theta_i \leq m$.

B. MC-Fluid Scheduling Algorithm

The fluid scheduling algorithm with a single static execution 
rate per task is inefficient in MC systems. If we assign 
$\theta_i := \frac{u^H_i}{t_i}$ for each HI task $\tau_i \in \tau_H$ and $\theta_i := \frac{u^L_i}{t_i}$ 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.
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 ($\theta_L^I$ and $\theta_H^I$) for each task $\tau_i \in \tau$. For a job $J_i^k$ of a task $\tau_i$, MC-Fluid assigns $\theta_L^I$ to $\tau_i(r_k^I, \min(t_M, d_k^I))$ and $\theta_H^I$ to $\tau_i(\max(t_M, r_k^I), d_k^I)$ where $r_k^I$ is its release time, $d_k^I$ is its deadline, and $t_M$ is the time instant of mode-switch. Since all LO-tasks are dropped at mode-switch, $\theta_H^I$ is not specified for all LO-tasks $\forall \tau_i \in \tau_L$.

Informally, MC-Fluid executes each task $\tau_i \in \tau$ with $\theta_L^I$ in LO-mode and with $\theta_H^I$ 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.** A task $\tau_i \in \tau$ in a time interval of length $t$ in LO- and HI-mode, denoted by $E_L^I(t)$ and $E_H^I(t)$, are the total amount of processor resources that the job has consumed during this time interval in LO- and HI-mode, respectively: $E_L^I(t) \equiv \theta_L^I \cdot t$ and $E_H^I(t) \equiv \theta_H^I \cdot t$.

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 $\tau$, where each task $\tau_i \in \tau$ has LO- and HI-execution rates ($\theta_L^I$ and $\theta_H^I$), is MC-schedulable under MC-Fluid iff
\[ \forall \tau_i \in \tau, \theta_L^I \geq u_L^I, \]
\[ \forall \tau_i \in \tau_H, u_L^I + u_H^I - u_L^I \cdot \theta_H^I \leq 1, \]
\[ \sum_{\tau_i \in \tau} \theta_L^I \leq m, \]
\[ \sum_{\tau_i \in \tau_H} \theta_H^I \leq m. \]

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 $\theta_L^I \geq u_L^I$.

**Proof:** ($\Rightarrow$) 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 ($C_L^I$). From $\theta_L^I \geq u_L^I$,
\[ \theta_L^I \cdot T_i \geq u_L^I \cdot T_i \Rightarrow E_L^I(T_i) \geq C_L^I. \] (by Def. 4)

($\Leftarrow$) We will prove the contrapositive: if $\theta_L^I < u_L^I$, then the task cannot meet its deadline in LO-mode. It is true because $E_L^I(T_i) = \theta_L^I \cdot T_i < u_L^I \cdot T_i = C_L^I$.

**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 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 $\tau_i$ 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 $\tau_i$, 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 $\theta_L^I$ before mode-switch and with $\theta_H^I$ 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 $\theta_L^I$ and its execution amount from mode switch to its deadline with $\theta_H^I$:
\[ E_L^I(w_i) + E_H^I(T_i - w_i) = \theta_L^I \cdot w_i + \theta_H^I \cdot (T_i - w_i) \]
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 ($C_H^I$):
\[ \theta_L^I \cdot w_i + \theta_H^I \cdot (T_i - w_i) \geq C_H^I. \] (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 $\forall w_i \in [0, T_i]$:
\[ \forall w_i, \theta_L^I \cdot w_i + \theta_H^I \cdot (T_i - w_i) \geq C_H^I. \] (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 \( \theta^L_i > \theta^H_i \) or \( \theta^L_i \leq \theta^H_i \). Lemma 3 considers the first case (\( \theta^L_i > \theta^H_i \)).

**Lemma 3.** A HI-task \( \tau_i \) with \( \theta^L_i > \theta^H_i \) can meet its deadline in HI-mode iff it meets its deadline in HI-mode when \( \theta^L_i = \theta^H_i \).

**Proof:** (\( \Leftarrow \)) Let \( \theta^L_i \) be the value of the original \( \theta^L_i \) where \( \theta^L_i > \theta^H_i \). Since the task can meet its deadline in HI-mode when \( \theta^L_i = \theta^H_i \), from Eq. (6), we have
\[
\forall w_i, \quad \theta^H_i \cdot w_i + \theta^H_i \cdot (T_i - w_i) = \theta^H_i \cdot T_i \geq C^H_i . \quad (7)
\]
To show that the task can meet its deadline in HI-mode when \( \theta^L_i = \theta^H_i \), we need to show that Eq. (6) holds: \( \forall w_i, \)
\[
\theta^L_i \cdot w_i + \theta^H_i \cdot (T_i - w_i) > \theta^H_i \cdot w_i + \theta^H_i \cdot (T_i - w_i) = \theta^H_i \cdot T_i ,
\]
which is greater than or equal to \( C^H_i \) by Eq. (7).

(\( \Rightarrow \)) Suppose that the task can meet its deadline in HI-mode. Then, we claim that \( \theta^H_i \geq w_i \). We prove it by contradiction: suppose that \( \theta^H_i < w_i \). Then, Eq. (6) does not hold when \( w_i = 0 \):
\[
\theta^L_i \cdot 0 + \theta^H_i \cdot (T_i - 0) = \theta^H_i \cdot T_i ,
\]
which is smaller than \( C^H_i \) because \( \theta^H_i \cdot T_i < \theta^H_i \cdot T_i = C^H_i \). 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 (\( \theta^H_i \geq w_i \)).

Next, we need to show that the task can meet its deadline in HI-mode when \( \theta^L_i = \theta^H_i \). Then, it is required to show that Eq. (6) holds:
\[
\forall w_i, \quad \theta^H_i \cdot w_i + \theta^H_i \cdot (T_i - w_i) = \theta^H_i \cdot T_i ,
\]
which is greater than or equal to \( C^H_i \) because \( \theta^H_i \geq w_i \).

Using below corollary, we assume that \( \theta^L_i \leq \theta^H_i \) for any task \( \tau_i \in \tau_H \) in the rest of the paper.

**Corollary 4.** For task-schedulability of a HI-task \( \tau_i \) in HI-mode, we only need to consider the case where \( \theta^L_i \leq \theta^H_i \).

**Proof:** Task-schedulability of the task in HI-mode when \( \theta^L_i > \theta^H_i \) is equivalent to the one when \( \theta^L_i = \theta^L_i \) by Lemma 3. Thus, its task-schedulability in HI-mode is equivalent to the one when \( \theta^L_i \leq \theta^H_i \).

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 \( \tau_i \) satisfying task-schedulability in LO-mode, the task can meet its deadline in HI-mode iff
\[
u^L_i \theta^L_i + u^H_i - u^L_i \theta^H_i \leq 1 .
\]

**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 instant cannot exceed its LO-WCET \( (C^H_i)^4 \). Thus, its execution amount from its release time to mode-switch also cannot exceed \( C^H_i \):
\[
E_i^L(w_i) \leq C^H_i \Rightarrow \theta^L_i \cdot w_i \leq C^H_i . \quad (8)
\]
Combining \( 0 \leq w_i \leq T_i \) and Eq. (8), we have \( 0 \leq w_i \leq \min(C^H_i/\theta^L_i, T_i) \).

Since task-schedulability in HI-mode holds, we have \( \theta^L_i \geq w_i \) by Lemma 2. Then,
\[
\theta^L_i \geq C^H_i/\theta^L_i \Rightarrow T_i \geq C^H_i/\theta^L_i . \quad \text{(multiplying by } T_i/\theta^L_i)
\]
Thus, the range of valid \( w_i \) is \( 0 \leq w_i \leq C^H_i/\theta^L_i \).

We know that task-schedulability in HI-mode is Eq. (6), which is rewritten to:
\[
\forall w_i, \quad \theta^L_i \cdot w_i + \theta^H_i \cdot (T_i - w_i) \geq C^H_i
\]
\[
\Leftrightarrow \forall w_i, \quad \theta^L_i \cdot w_i + \theta^H_i \cdot (T_i - w_i) > \theta^H_i \cdot w_i + \theta^H_i \cdot (T_i - w_i) = \theta^H_i \cdot T_i ,
\]
\[
\Leftrightarrow \theta^H_i \cdot T_i \geq (\theta^H_i - \theta^L_i) \cdot w_i + C^H_i \quad \text{and} \quad \theta^H_i \cdot T_i \geq C^H_i/\theta^L_i ,
\]
\[
\Rightarrow \theta^H_i \cdot T_i \geq C^H_i/\theta^L_i + C^H_i - C^H_i = C^H_i . \quad (\text{dividing by } \theta^H_i \cdot T_i)
\]
Thus, the task can meet its deadline in HI-mode iff Eq. (6) holds iff \( 1 \geq u^L_i/\theta^L_i + (u^H_i - u^L_i)\theta^H_i \).

**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 \( C^H_i \) at mode-switch. On fluid platforms, this situation happens because all tasks are always executed with their execution rates whenever they are ready.

**Example 1.** Consider a two-processor system where its task set \( \tau \) 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

<table>
<thead>
<tr>
<th>( \tau_i )</th>
<th>( C^L_i )</th>
<th>( C^H_i )</th>
<th>( x_i )</th>
<th>( u^L_i )</th>
<th>( u^H_i )</th>
<th>( \theta^L_i )</th>
<th>( \theta^H_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \tau_1 )</td>
<td>10</td>
<td>3</td>
<td>HI</td>
<td>0.5</td>
<td>0.8</td>
<td>6/10</td>
<td>1</td>
</tr>
<tr>
<td>( \tau_2 )</td>
<td>20</td>
<td>8</td>
<td>HI</td>
<td>0.4</td>
<td>0.7</td>
<td>6/10</td>
<td>9/10</td>
</tr>
<tr>
<td>( \tau_3 )</td>
<td>30</td>
<td>3</td>
<td>HI</td>
<td>0.1</td>
<td>0.3</td>
<td>1/10</td>
<td>1/10</td>
</tr>
<tr>
<td>( \tau_4 )</td>
<td>40</td>
<td>20</td>
<td>LO</td>
<td>0.5</td>
<td>0.5</td>
<td>5/10</td>
<td>5/10</td>
</tr>
</tbody>
</table>

\footnote{4}{The job triggers mode-switch if the job execute for more than \( C^H_i \).}

\footnote{5}{Note that we already assumed that \( \theta^L_i \leq \theta^H_i \) by Corollary 4.}
easily check that Eq. (1) holds. We show that Eq. (2) holds: 
\[
\frac{0.3}{6/10} + \frac{0.8 - 0.3}{1} = 1 \text{ for } \tau_1, \quad \frac{0.4}{6/10} + \frac{0.7 - 0.4}{1} = 1 \text{ for } \tau_2, \quad \text{and } \frac{0.1}{10/10} + 0 \leq 1 \text{ for } \tau_3.
\]
We can check that \(\sum_{\tau_i \in \tau_i} \theta_i^L = \frac{6 + 6 + 1 + 5}{10} = 2\) for Eq. (3) and \(\sum_{\tau_i \in \tau_i} \theta_i^H = \frac{10 + 2}{10} = 2\) 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 \(\tau\) 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 \(\lambda\) is called **optimal** if \(\lambda\) 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 \(\theta^L\) and \(\theta^H\), respectively.

**Lemma 6** (An optimal assignment of \(\theta^L\)). If a task set \(\tau\) 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) \(\theta_i^L := \frac{u_i^L - \theta_i^H}{\theta_i^H - u_i^L + u_i^H}\) for \(\tau_i \in \tau_H\).

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

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

Let \(\theta_i^L\) be the value of \(\theta_i^L\) in \(\lambda\). Since \(\lambda\) is schedulable, Eq. (1) holds with \(\theta_i^L\), 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 \(\theta_i^L\) (from \(\theta_i^L\) 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 (\(\theta_i^H\)) in \(\lambda\). We will show that \(\tau_i\) is still schedulable after reassignment of \(\theta_i^H\). If we reassign \(\theta_i^H\), it only affects Eqs. (1), (2), and (3).

Let \(\theta_i^L\) and \(\theta_i^H\) be the value of \(\theta_i^L\) and \(\theta_i^H\) in \(\lambda\), respectively. Since Eq. (2) holds for \(\theta_i^L\) and \(\theta_i^H\) in \(\lambda\), we have
\[
\frac{u_i^H - u_i^L}{\theta_i^H} \leq 1 - \frac{u_i^H - u_i^L}{\theta_i^H} = \frac{u_i^H - u_i^L}{\theta_i^H} < 1,
\]
which is Eq. (9).

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

**Theorem 2** (Conditions for an optimal assignment of \(\theta_i^H\)). A task set \(\tau\) is feasible if there exists an assignment of \(\theta_i^L\) for \(\forall \tau_i \in \tau_H\) satisfying
\[
\begin{align*}
\theta_i^H &\leq \frac{u_i^H}{\theta_i^H} \leq \theta_i^H, & (10) \\
U_L^H + U_H^L + \sum_{\tau_i \in \tau_H} \frac{(u_i^H - u_i^L)}{\theta_i^H} &\leq m, & (11) \\
\sum_{\tau_i \in \tau_H} \theta_i^H &\leq m. & (12)
\end{align*}
\]

**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 \(\tau_i \in \tau_L\), \(\theta_i^H := \frac{u_i^L \cdot \theta_i^H}{u_i^L + u_i^H}\) for each task \(\tau_i \in \tau_H\), and \(\theta_i^H\) 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 \(\theta_i^L\) for \(\forall \tau_i \in \tau\) satisfies Eq. (1) and \(\theta_i^H\) for \(\forall \tau_i \in \tau_H\) satisfies Eq. (2). We can rewrite Eq. (3) to:
\[
\begin{align*}
\sum_{\tau_i \in \tau} \theta_i^L &\leq m \iff \sum_{\tau_i \in \tau_L} \theta_i^L + \sum_{\tau_i \in \tau_H} \theta_i^L \leq m \iff \\
\sum_{\tau_i \in \tau_L} u_i^L + \sum_{\tau_i \in \tau_H} \frac{u_i^L \cdot \theta_i^H}{\theta_i^H - u_i^L + u_i^H} &\leq m \iff \\
U_L^H + \sum_{\tau_i \in \tau_H} \left( \frac{u_i^L \cdot (u_i^H - u_i^L)}{\theta_i^H - u_i^L + u_i^H} \right) &\leq m,
\end{align*}
\]
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 \(\theta_i^H := \frac{u_i^L \cdot \theta_i^H}{u_i^L + u_i^H}\) by Lemma 6. We need to show that \(\theta_i^H\) for \(\forall \tau_i \in \tau_H\) in the assignment satisfies Eqs. (10), (11), and (12).

Consider \(\theta_i^H\) 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 \(\theta_i^H \leq \theta_i^H\): to:
\[
\frac{u_i^L \cdot \theta_i^H}{\theta_i^H - u_i^L + u_i^H} \leq \theta_i^H \iff u_i^L \leq \theta_i^H - u_i^H + u_i^L,
\]
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 \(\theta_i^H\) by Lemma 6. Although we have Theorem 2, an assignment algorithm cannot easily determine \(\theta_i^H\) 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 \(\theta_i^H\) by Def. 6 is optimal according to Def. 5.
Definition 6 (Assignment Problem). Given a task set \( \tau \), we define a non-negative real number \( X_i \) for each task \( \tau_i \in \tau \) such that \( \theta_i^\tau := u_i^\tau + X_i \) and \( X_i \) is the solution to the following optimization problem:

\[
\begin{align*}
\text{minimize} & \quad \sum_{\tau_i \in \tau} \frac{u_i^L (u_i^H - u_i^L)}{X_i + u_i^L} \\
\text{subject to} & \quad \sum_{\tau_i \in \tau} X_i - m + u_H^\tau \leq 0, \quad \text{(CON1)} \\
& \quad \forall \tau_i \in \tau_H, \quad -X_i \leq 0, \quad \text{(CON2)} \\
& \quad \forall \tau_i \in \tau_H, \quad X_i - 1 + u_H^\tau \leq 0. \quad \text{(CON3)}
\end{align*}
\]

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

Proof: We claim that if an assignment by Def. 6 cannot satisfy Eqs. (10), (11), and (12), no assignment can satisfy those equations. Then, \( \tau \) 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 \( \theta_i^\tau := u_i^H + X_i^* \) to \( \forall \tau_i \in \tau_H \). Then, Eq. (10) holds from CON2 and CON3 and Eq. (12) holds from CON1. Suppose that Eq. (11) does not hold with the assignment, meaning OBJ* > \( m - U_L^H - U_H^L \). For any set of \( X_i \) satisfying CON1, CON2, and CON3, we have \( \sum_{\tau_i \in \tau_H} \frac{u_i^L (u_i^H - u_i^L)}{X_i + u_i^L} \geq \text{OBJ*} \) by the definition of the optimization problem and thus Eq. (11) does not hold:

\[
\sum_{\tau_i \in \tau_H} \frac{u_i^L (u_i^H - u_i^L)}{X_i + u_i^L} \geq \text{OBJ*} > m - U_L^H - U_H^L.
\]

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

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 \( \bar{\tau} \) is a vector of \( x \) for \( i = 1, \cdots, n \) where \( n = |\bar{\tau}| \) and \( x_i^* \) be the value of \( x_i \). Consider an optimization problem:

\[
\begin{align*}
\text{minimize} & \quad f(\bar{x}) \text{ subject to } g_j(\bar{x}) \leq 0 \text{ for } j = 1, \cdots, N \\
\end{align*}
\]

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

\[
\begin{align*}
\forall j, & \quad g_j(\bar{x}^*) \leq 0, \quad (13) \\
\forall j, & \quad \lambda^*_j \cdot g_j(\bar{x}^*) = 0, \quad (14) \\
\forall i, & \quad \frac{\partial f(\bar{x})}{\partial x_i} + \sum_j (\lambda^*_j \frac{\partial g_j(\bar{x})}{\partial x_i}) = 0, \quad (15) \\
\forall j, & \quad \lambda^*_j \geq 0, \quad (16)
\end{align*}
\]

where \( \lambda_j \) is a Lagrange multiplier, \( \lambda^*_j \) is the value of \( \lambda_j \), and \( \bar{x}^* \) is a vector of \( \lambda_j \).

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 \( \psi, \lambda_i, \) and \( \nu_i \) as Lagrange multipliers for CON1, CON2, and CON3, respectively. We denote a vector of \( X_i, \lambda_i, \) and \( \nu_i \) by \( X, \lambda, \) and \( \nu \), and denote the value of \( X_i, \lambda_i, \) and \( \nu_i \) by \( X_i^*, \lambda_i^*, \) and \( \nu_i^* \), respectively. We define, for each task \( \tau_i \),

\[
\text{Cost}_i(x) \overset{\text{def}}{=} \frac{u_i^L (u_i^H - u_i^L)}{(x + u_i^L)^2} \text{ where } x \in \mathbb{R}^+.
\]

Lemma 9. Consider the optimization problem in Def. 6. \( X^* \) minimizes \( f(X) \) iff there exist \( \psi^*, X^*, \) and \( \nu^* \) s.t.

\[
\begin{align*}
\sum_{\tau_i} X_i^* - m + U_H^H \leq 0, \quad (17) \\
\forall \tau_i, & \quad -X_i^* \leq 0, \quad X_i^* - 1 + u_H^H \leq 0, \quad (18) \\
\psi^*(\sum_{\tau_i} X_i^* - m + U_H^H) = 0, \quad (19) \\
\forall \tau_i, & \quad \lambda_i^*(-X_i^* - 1 + u_H^H) = 0, \quad (20) \\
\forall \tau_i, & \quad -\psi^*(X_i^*) + \psi^* - \lambda_i^* + \nu_i^* = 0, \quad (21) \\
\psi^* \geq 0 \land \forall \tau_i (\lambda_i^* \geq 0 \land \nu_i^* \geq 0), \quad (22)
\end{align*}
\]

where \( f(X) = \sum_{\tau_i} u_i^L (u_i^H - u_i^L) \).

Proof: We will derive KKT conditions for the optimization problem in Def. 6: \( f(X) = \sum_{\tau_i} \frac{u_i^L (u_i^H - u_i^L)}{X_i + u_i^L} \) and \( g_1(X) = \sum_{\tau_i} X_i - m + U_H^H \) for CON1; for \( \forall \tau_i \), \( g_2(X) = -X_i \) for CON2 and \( g_3(X) = X_i - 1 + u_H^H \) for CON3.

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

\[
\forall \tau_i, \forall X_i, \quad \frac{\partial^2 f(X)}{\partial (X_i)^2} = \frac{2 \cdot u_i^L (u_i^H - u_i^L)}{(X_i + u_i^L)^3} \geq 0 \text{ and }
\]

\[
\forall \tau_i, \forall \tau_j, \forall X_i, \quad \frac{\partial^2 g_1(X)}{\partial (X_i)^2} = \frac{\partial^2 g_2(X)}{\partial (X_i)^2} = \frac{\partial^2 g_3(X)}{\partial (X_i)^2} = 0.
\]

We derive KKT conditions for the problem by Lemma 8:

- We denote Lagrange multipliers for \( g_1(X), g_2(X), \) and \( g_3(X) \) by \( \psi, \lambda_i, \) and \( \nu_i \).
- Eq. (13) for \( g_1(X) \) is Eq. (17). Eq. (13) for \( g_2(X) \) is Eq. (18).
- Eq. (14) for \( g_1(X) \) is Eq. (19). Eq. (14) for \( g_2(X) \) and \( g_3(X) \) is Eq. (20).
• Eq. (15) is \( \forall \tau_i, \frac{\partial f(\bar{x})}{\partial x} \psi + \lambda_i + \nu_i = 0 \), which is Eq. (21).
• Eq. (16) for \( g_1(\bar{x}), g_2(\bar{x}) \) and \( g_3(\bar{x}) \) is Eq. (22).

By Lemma 8, \( X^* \) minimizes \( f(\bar{x}) \) if there exist \( \psi^*, X^* \), and \( \nu^* \) satisfying Eqs. (17), (18), (19), (20), (21), and (22). \( \blacksquare \)

Example 2. Consider the system in Example 1. We need to solve the optimization problem in Def. 6. Suppose that we have \( X^* \) satisfying KKT conditions in Lemma 9 for the system: \( X_1^* = 0.2, X_2^* = 0.2, \) and \( X_3^* = 0 \). Eq. (17) holds: \( 0.2 + 0.2 - 0 + 1.6 = 0 \). We can easily check that Eqs. (18) and (19) hold. To satisfy Eq. (20), we have \( \lambda_1^* = \lambda_2^* = \lambda_3^* = 0 \). By Eq. (21), we have \( \frac{0.12}{5} + \nu_2^* = 0, \) \( \frac{0.12}{5} + \nu_3^* = 0, \) and \( \nu_3^* = 0 \). Then, we have \( \psi^* = 1/3, \nu_2^* = 4/15, \) and \( \nu_3^* = 1/3 \). Eq. (22) holds with \( \psi^*, X^* \), and \( \nu^* \). Thus, \( X^* \) is the solution to the problem in Def. 6 by Lemma 9.

By Def. 6, we have \( \theta^H_1 := 1, \theta^H_2 := 0.9, \) and \( \theta^H_3 := 0.1 \). We can assign \( \theta^L_i \) 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 \( \lambda_i^* \) and \( \nu_i^* \) for each task \( \tau_i \) depend only on \( X^*_i \). Since only \( \psi^* \) depends on \( X^* \), if we know \( \psi^* \) satisfying our KKT conditions, we can independently analyze \( \lambda_i^* \), \( \nu_i^* \), and \( X^*_i \) for each task \( \tau_i \). 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 \( \tau_i \) and \( \psi \geq 0 \), if we assign \( X_i := \text{Cal}_X(\psi) \), then Eqs. (18), (20), and (21) hold with some \( \lambda_i \) and \( \nu_i \).

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 \( \text{Cal}_X(\psi) \).

Case \( \psi \geq \text{Cost}_i(0) \). For some \( \nu_i \geq 0 \), Eq. (21) is simplified to:
\[
\nu_i \leq \text{Cost}_i(X_i) + \lambda_i,
\]
by removing \( \nu_i \). Note that \( \text{Cost}_i(X_i) \leq \text{Cost}_i(0) \) where \( 0 \leq X_i \leq 1 - u_i^H \). To satisfy Eq. (23), \( \lambda_i \) should satisfy that \( \psi - \text{Cost}_i(X_i) \geq \psi - \text{Cost}_i(0) \), 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 \( \psi < \text{Cost}_i(1 - u_i^H) \). For some \( \lambda_i \geq 0 \), Eq. (21) is simplified to:
\[
\psi + \nu_i \geq \text{Cost}_i(X_i),
\]
by removing \( \lambda_i \). Note that \( \text{Cost}_i(1 - u_i^H) \leq \text{Cost}_i(X_i) \) where \( 0 \leq X_i \leq 1 - u_i^H \). To satisfy Eq. (24), \( \nu_i \) should satisfy that \( \nu_i \geq \text{Cost}_i(X_i) - \psi \geq \text{Cost}_i(1 - u_i^H) - \psi \), which is greater than 0 from the assumption. To satisfy Eq. (20), we assign \( \nu_i := 0 \) and \( X_i := 1 - u_i^H \). Thus, Eqs. (20) and (21) hold.

Case \( \text{Cost}_i(1 - u_i^H) \leq \psi < \text{Cost}_i(0) \). To satisfy Eq. (20), we assign \( \nu_i := 0 \) and \( X_i := 0 \). Then, to satisfy Eq. (21), we need to satisfy \( \psi = \text{Cost}_i(X_i) \), from which, \( X_i \) can be computed:
\[
\psi = \frac{u_i^H - u_i^L}{2} + \frac{u_i^H - u_i^L}{2} \leq \frac{u_i^H - u_i^L}{2} \Rightarrow X_i = \sqrt{\frac{u_i^H - u_i^L}{2} \psi} - u_i^L.
\]

Combining three cases, we showed that Eqs. (18), (20), and (21) hold with some positive \( \lambda_i \) and \( \nu_i \).

To find \( \psi \) satisfying our KKT condition, we divide the range of \( \psi \). According to Lemma 10, if \( \psi \) is greater than or equal to \( \text{Cost}_i(0) \) or smaller than \( \text{Cost}_i(1 - u_i^H) \), \( \text{Cal}_X(\psi) \) is a constant; otherwise, \( \text{Cal}_X(\psi) \) is a linear function of \( 1/\sqrt{\psi} \).

To divide the range of \( \psi \), we define an ordered set \( G_i := \{ x \in \mathbb{R}^+ \mid x = \text{Cost}_i(0) \} \) or \( \text{Cost}_i(1 - u_i^H) \) for \( \forall \tau_i \}, \) sorted in increasing order. We denote the \( j \)-th element of \( G \) as \( G \). In OERA, we utilize the property that \( \text{Cal}_X(\psi) \) for some task \( \tau_i \) 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 \( \psi^* \) satisfying our KKT conditions is greater than zero. We assign \( X_i := \text{Cal}_X(\psi^*) \) 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 \( \sum_{\tau_i} X_i - m + U_i^H = 0 \), which is \( \sum_{\tau_i} \text{Cal}_X(\psi^*) = m - U_i^H \).

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 \( \psi^* \) where some \( \psi^* \) 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 \( \psi^* \) by solving the condition (Line 3) and OERA returns \( X^* \) where \( X^*_i = \text{Cal}_X(\psi^*) \) (Line 4).

However, it is possible that there does not exist positive \( \psi^* \) satisfying our KKT conditions, which means that \( \psi^* \) satisfying our KKT conditions is zero because it is non-negative. OERA returns \( X^* \) where \( X^*_i = \text{Cal}_X(0) \) (Line 7). Theorem 3 proves the correctness of OERA.

Algorithm 1 The OERA algorithm

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

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 \( \sum_{\tau_i} \text{Cal}_X(\psi) \), 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 \( m - U_i^H \). When
value of \(\psi\). If it holds, the algorithm can find a unique solution of conditions in Lemma 9.

**Theorem 3.** Algorithm 1 (OERA) can find \(X^\tau\) 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 \(X_i := \text{Cal}_i(\psi^*)\) for \(\forall i\). By Lemma 10, Eqs. (18), (20), (21), and (22) hold. To satisfy Eqs. (17) and (19), we require the condition \(\sum_i \text{Cal}_i(\psi^*) = m - U_H^i\), which is not greater than \(m - U_H^i\) from the assumption. Thus, the required condition holds.

Since the KKT conditions hold with \(\psi^* = 0\), the algorithm returns \(X^\tau\) where \(X^\tau_i = \text{Cal}_i(\psi^*)\) satisfying the KKT conditions.

**Algorithm Complexity.** Let \(n = |\tau|\). Algorithm 1 (OERA) can find a solution at most \(O(n)\) iteration because \(|G| \leq 2|\tau_H| \leq 2n\). 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 \(A\) is defined as a real number \(\alpha (\geq 1)\) such that any task set schedulable by an optimal clairvoyant algorithm\(^6\) on \(m\) speed-1 processors is also schedulable by \(A\) on \(m\) speed-\(\alpha\) processors. In other words, a task set that is clairvoyantly schedulable on \(m\) speed-(1/\(\alpha\)) processors is also schedulable by \(A\) on \(m\) speed-\(\alpha\) processors. We consider this special task set (clairvoyantly schedulable on \(m\) speed-(1/\(\alpha\))) for speedup factor derivation.

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

\[
0 \leq x \leq 1, \\
U_L^i + U_H^i + \frac{U_H^i - U_L^i}{(\alpha - 1)x + 1} \leq m, \\
U_L^i + (\alpha - 1)U_H^i : x \leq m.
\]

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

Since \(\tau\) is clairvoyantly schedulable on \(m\) speed-(1/\(\alpha\)) processors, we assume that \(U_L^i + U_H^i \leq m/\alpha\) and \(U_H^i \leq m/\alpha\).

Since the task is assumed to be executable on a speed-(1/\(\alpha\)) processor, we assume that \(u_L^i \leq u_H^i \leq 1/\alpha\) for \(\forall \tau_i \in \tau_H\).

Let \(\lambda\) be an assignment where \(\theta_H^i := u_H^i + (\alpha - 1)u_L^i \cdot x\) for \(\forall \tau_i \in \tau_H\) satisfying Eq. (25), (26), and (27). We show that the assigned \(\theta_H^i\) by \(x\) is no greater than 1 from the definition of the execution rate (Def. 2), for \(\forall \tau_i \in \tau_H\).

\[
\theta_H^i = u_H^i + (\alpha - 1)u_L^i \cdot x \\
\leq u_H^i + (\alpha - 1)u_L^i \\
\leq u_H^i + (\alpha - 1)u_H^i \\
\leq 1.
\]

\(\text{In MC systems, a clairvoyant (fluid or non-fluid) scheduling algorithm is the one that knows the time instant of mode-switch before runtime scheduling.}\)
We show that $A$ satisfies Eq. (10): from Eq. (25), for $\forall \tau_i \in \tau_H$, 
\[
0 \leq (\alpha - 1)u_i^L \cdot x \Rightarrow u_i^H \leq u_i^H + (\alpha - 1)u_i^L \cdot x \\
\Rightarrow u_i^H \leq \theta_i^H,
\]
which is Eq. (10).

(ii) We show that $A$ satisfies Eq. (11): from Eq. (26), 
\[
U_i^L + U_i^L + \frac{U_i^H - U_i^L}{(\alpha - 1) \cdot x + 1} \leq m
\]
\[
\Rightarrow U_i^L + U_i^L + \sum_{\tau_i \in \tau_H} \frac{u_i^H - u_i^L}{(\alpha - 1) \cdot x + 1} \leq m,
\]
\[
\Rightarrow U_i^L + U_i^L + \sum_{\tau_i \in \tau_H} \frac{\theta_i^H - u_i^H + u_i^L}{x} \leq m,
\]
\[
\therefore (\alpha - 1)u_i^L \cdot x + u_i^L = \theta_i^H - u_i^H + u_i^L
\]
which is Eq. (11).

(iii) We show that $A$ satisfies Eq. (12): from Eq. (27), 
\[
U_i^H + (\alpha - 1)U_i^L \cdot x \leq m \Rightarrow \sum_{\tau_i \in \tau_H} (u_i^H + (\alpha - 1)u_i^L \cdot x) \leq m
\]
\[
\Rightarrow \sum_{\tau_i \in \tau_H} \theta_i^H \leq m,
\]
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:
\[
U_i^L + U_i^L + \frac{U_i^H - U_i^L}{(\alpha - 1) \cdot x + 1} \leq m
\]
\[
\Rightarrow \frac{U_i^H - U_i^L}{(\alpha - 1) \cdot x + 1} \leq m - U_i^L - U_i^L
\]
\[
\Rightarrow \frac{U_i^H - U_i^L}{m - U_i^L - U_i^L} \leq (\alpha - 1) \cdot x + 1
\]
\[
\Rightarrow m - U_i^L - U_i^L \leq (\alpha - 1) \cdot x
\]
\[
\Rightarrow \frac{U_i^H + U_i^L - m}{(\alpha - 1)(m - U_i^L - U_i^L)} \leq x.
\]
\[
\therefore (\alpha - 1) > 0
\]  
\[
\text{Eq. (28)}
\]
Since LHS of Eq. (28) is non-negative, the lower bound of $x$ is 
\[
\frac{m - U_i^H}{(\alpha - 1)(m - U_i^L - U_i^L)}
\]
by Eqs. (25) and (28).

We will find the upper bound of the feasible $x$. Eq. (27) can be rewritten to $x \leq \frac{m - U_i^H}{(\alpha - 1)U_i^L}$. Since $(\alpha - 1)U_i^L + U_i^H \leq (\alpha - 1)U_i^H + U_i^H \leq m$, we have 
\[
(\alpha - 1)U_i^L + U_i^H \leq m \Rightarrow 1 \leq \frac{m - U_i^H}{(\alpha - 1)U_i^L}.
\]  
\[
\text{Eq. (29)}
\]
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):
\[
\frac{U_i^H + U_i^L - m}{(\alpha - 1)(m - U_i^L - U_i^L)} \leq 1.
\]
\[
\text{Eq. (29)}
\]
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-$l$ 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:
\[
U_i^H + U_i^L - m \leq (\alpha - 1)(m - U_i^L - U_i^L)
\]
\[
\Rightarrow U_i^H + (\alpha - 1)U_i^L + \alpha \cdot U_i^L \leq \alpha \cdot m
\]
\[
\Rightarrow m/\alpha + (\alpha - 1)(m/\alpha - U_i^L) + \alpha \cdot U_i^L \leq \alpha \cdot m
\]
\[
\Rightarrow m + U_i^L \leq \alpha \cdot m
\]
\[
\Rightarrow m + m/\alpha \leq \alpha \cdot m.
\]
\[
\therefore U_i^L \leq U_i^L + U_i^L \leq m/\alpha
\]  
\[
\text{Eq. (30)}
\]
Eq. (30) holds when $\alpha \geq (1 + \sqrt{5})/2$ because Eq. (30) is $\alpha^2 - \alpha - 1 \geq 0$ and $\alpha$ 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 $\tau_i$ within a time slice of interval length $l$ is calculated as $l \cdot \delta_i$. 

\[
\text{Algorithm}
\]  
\[
\text{Theorem 4}
\]  
\[
\text{Proof}
\]  
\[
\text{VII. MC-DP-Fair Scheduling Algorithm}
\]  
\[
\text{DP-Fair}
\]  
\[
\text{Required execution amount for a job of a non-MC task $\tau_i$ within a time slice of interval length $l$ is calculated as $l \cdot \delta_i$.}
\]
where $\delta_i$ 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 $\tau$ and a time slice, if the task set is scheduled within the time slice under DP-Fair and $\sum_{i \in \tau} \delta_i \leq m$, 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 $\tau$ is schedulable under DP-Fair iff $\sum_{\tau_i \in \tau} \delta_i \leq m$.

**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 considered to ensure the schedulability properties of DP-Fair. The scheduling algorithm, which constructs a non-fluid schedule, is based on the following lemmas.

**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. $0 < V_i \leq T_i$. Let $\Gamma$ 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 $\Gamma$ and with $T_i$ after $\Gamma$ as its deadline.

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

In LO-mode, MC-DP-Fair considers, for a task $\tau_i \in \tau$, $\delta^*_L \triangleq C_i^L/V_i$, where $V_i$ is the virtual deadline of $\tau_i$. Then, the amount of execution required for a task $\tau_i$ and time slice length $l$ is $l \cdot \delta^*_L$ 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 only need to consider the first case. Consider that mode-switch happens in the middle of a time slice. Note that $\Gamma$ indicates the end of this time slice. MC-DP-Fair executes HI-tasks for the amount of their required remaining execution (calculated based on $C_i^H$) until $\Gamma$. Then, the second case is equivalent to the case where mode-switch happens at $\Gamma$, 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 $\sum_{\tau_i \in \tau} \delta^*_L \leq m$, we can compute the execution amount of the job from its release time to the DP: $E_i^L(w_i) = \delta^*_L \cdot w_i$. We can compute $\delta^*_H$ after the DP:

$$\delta^*_H \triangleq \frac{C_i^H - E_i^L(w_i)}{T_i - w_i} = \frac{C_i^H - \delta^*_L \cdot w_i}{T_i - w_i}. \quad (31)$$

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

**Virtual Deadline Assignment.** In this section, we denote the values of $\theta^*_L$ and $\theta^*_H$ in the optimal assignment of MC-Fluid (by $\theta^*_L$ and $\theta^*_H$), respectively. Intuitively, to utilize MC-Fluid analysis, the required LO-density for a HI-task is no greater than $\delta^*_L$ 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 $V_i := C_i^L/\theta^*_L$ for a HI-task $\tau_i \in \tau_H$ where $\theta^*_L$ is the value of $\theta^*_L$ in the optimal assignment for MC-Fluid.

**Lemma 15.** Given a feasible MC task set $\tau$, if the task set is scheduled by MC-DP-Fair with the virtual deadline assignment by Def. 8, then (i) $\delta^*_L \leq \theta^*_L$ for each task $\tau_i \in \tau$ and (ii) $\delta^*_H \leq \theta^*_H$ for each task $\tau_i \in \tau_H$.

**Proof:** (i) We show that $\delta^*_L \leq \theta^*_L$ for $\tau_i \in \tau_L$: we have

$$\delta^*_L = \frac{C_i^L}{V_i} = \theta^*_L \quad \text{for} \quad \tau_i \in \tau_L \quad \text{by Lemma 6 and}$$

$$\delta^*_L = \frac{C_i^L}{V_i} = \theta^*_L \quad \text{for} \quad \tau_i \in \tau_H.$$

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

To find the maximum $\delta^*_H$, consider derivative of Eq. (31):

$$\frac{d}{dw_i} \delta^*_H = \frac{-\delta^*_L(T_i - w_i) + (C_i^H - \delta^*_L \cdot w_i)}{(T_i - w_i)^2} = \frac{C_i^H - \delta^*_L \cdot T_i}{(T_i - w_i)^2},$$

which is greater than or equal to 0 if $w_i = \theta^*_H$.

To show that the derivative is non-negative, we show that $w_i \geq \theta^*_H$. Since $\theta^*_L \geq \delta^*_L$ from Case (i), we only need to show $w_i \geq \theta^*_H$: since $\theta^*_L = \frac{u_i^L}{\sum_{i \in \tau_L} u_i^L}$ by Lemma 6,

$$u_i^H \geq \frac{u_i^L \cdot \theta^*_H}{\theta^*_H - u_i^L}, \quad \Rightarrow u_i^H(\theta^*_H - u_i^L + u_i^L) \geq u_i^L \cdot \theta^*_H,$$

$$\Rightarrow \theta^*_H(u_i^H - u_i^L) \geq u_i^H(u_i^I - u_i^L),$$

which is true because $\theta^*_H$ satisfies Eq. (10), which is $\theta^*_H \geq u_i^H$, by Theorem 2.

From the derivative of $\delta^*_H$, we can find the maximum $\delta^*_H$: the maximum $\delta^*_H$ can be found at the maximum $w_i$ because $\frac{d}{dw_i} \delta^*_H \geq 0$ and thus $\delta^*_H$ is an increasing function of $w_i$. Since the task sets are equal in LO-mode, we have $w_i \leq V_i$. Then, we can calculate $\delta^*_H$ when $w_i = V_i$:

$$\delta^*_H = \frac{C_i^H - \delta^*_L \cdot V_i}{T_i - V_i} = \frac{C_i^H - C_i^L/V_i \cdot V_i}{T_i - C_i^L/\theta^*_L} \leq \frac{u_i^H - u_i^L}{1 - u_i^L/\theta^*_L},$$

which is $\theta^*_H$ by Lemma 6. Thus, we conclude $\delta^*_H \leq \theta^*_H$. ■
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_L^i and δ_H^i for each task τ_i ∈ τ, a MC task set τ is MC-schedulable iff

\[ \sum_{τ_i ∈ τ} δ_L^i ≤ m, \]  
\[ \sum_{τ_i ∈ τ} δ_H^i ≤ m. \]  

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 if and only if τ is MC-Fluid-feasible.

Proof: (⇒) 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^H is for ∀τ_i ∈ τ_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:

\[ \sum_{τ_i ∈ τ} δ_L^i ≤ m \iff \sum_{τ_i ∈ τ_L} C_L^i / T_i + \sum_{τ_i ∈ τ_H} C_H^i / V_i ≤ m \]
\[ \iff U_L^i + \sum_{τ_i ∈ τ_H} \frac{u_L^i · θ_h^i}{θ_h^i - u_H^i + u_L^i} ≤ m \]
\[ \iff U_L^i + \sum_{τ_i ∈ τ_H} \left( u_L^i + \frac{u_L^i(u_H^i - u_L^i)}{θ_h^i - u_H^i + u_L^i} \right) ≤ m \]
\[ \iff U_L^i + U_H^i + \sum_{τ_i ∈ τ_H} \frac{u_L^i(u_H^i - u_L^i)}{θ_h^i - u_H^i + u_L^i} ≤ m, \]

which is Eq. (11).

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

Since the feasible task set is scheduled by MC-DP-Fair with Def. 8, we have δ^H ≤ θ^H for each task τ_i ∈ τ_H by Lemma 15. We show that Eq. (33) holds:

\[ \sum_{τ_i ∈ τ_H} δ_L^i ≤ \sum_{τ_i ∈ τ_H} θ_H^i ≤ m, \]

which is true because the optimal assignment satisfies CON1 in Def. 6 with θ^H = X_i + u_H^i.

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 random drawn in uniform distribution):

- u_L^i 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_H^i / u_L^i) 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 χ_i := LO and C_L^i := u_L^i · T_i. Otherwise, set χ_i := HI, C_L^i := u_L^i · T_i, and C_H^i := u_L^i · R_i · T_i.

Repeat to generate a task in the task set until max(U_L^i + U_L^i, U_H^i) 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 ∈ {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
algorithm, which has the same scheduling properties as MC-Fluid. We showed that MC-Fluid has a speedup factor of $(1 + \sqrt{3})/2 \approx 1.618$, 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.

ACKNOWLEDGEMENT

This work was supported in part by BSRP (NRF-2010-0006650, NRF-2012R1A1A1014930), NCRC (2012-0009980), IITP (2011-0041513, 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