EDF-VD Scheduling of Flexible Mixed-Criticality System With Multiple-Shot Transitions

Gang Chen, Nan Guan, Biao Hu, and Wang Yi, *Fellow, IEEE*

***Abstract*—The existing mixed-criticality (MC) real-time task models assume that once any high-criticality task overruns, *all* high-criticality jobs execute up to their most pessimistic WCET estimations simultaneously in a one-shot manner. This is very pessimistic in the sense of unnecessary resource overbooking. In this paper, we propose a more generalized mixed-critical real-time task model, called flexible MC model with multiple-shot transitions (FMC-MST), to address this problem. In FMC-MST, high-criticality tasks can transit multiple intermediate levels to handle less pessimistic overruns independently and to nonuniformly scale the deadline on each level. We develop a run-time schedulability analysis for FMC-MST under EDF-VD scheduling, in which a better tradeoff between the penalties of low-criticality tasks and the overruns of high-criticality tasks is achieved to improve the service quality of low-criticality tasks. We also develop a resource optimization technique to find resource-efficient level-insertion configurations for FMC-MST task systems under MC timing constraints. Experiments demonstrate the effectiveness of FMC-MST compared with the state-of-the-art techniques.**

***Index Terms*—EDF-VD scheduling, flexible mixed-criticality (FMC) system, multiple-shot transitions.**

Manuscript received April 3, 2018; revised June 8, 2018; accepted July 2, 2018. This work was supported in part by the National Natural Science Foundation of China under Grant 61702085, Grant 61532007, Grant 61672140, and Grant 61772123, in part by the Fundamental Research Funds for the Central Universities under Grant N161604002, in part by RGC of Hong Kong under Grant ECS-25204216 and Grant GRF-15204917, in part by the University Grants Committee of Hong Kong through Hong Kong Polytechnic University under Project 1-ZVJ2, and in part by the Ministry of Education Joint Foundation for Equipment Pre-Research under Grant 6141A020333. This article was presented in the International Conference on Embedded Software 2018 and appears as part of the ESWEEK-TCAD special issue. *(Corresponding author: Gang Chen.)*

G. Chen is with the Smart Systems Laboratory, School of Computer Science and Engineering, Northeastern University, Shenyang 110819, China, and also with the Department of Computing, Hong Kong Polytechnic University, Hong Kong (e-mail:chengang@cse.neu.edu.cn).

N. Guan is with the Department of Computing, Hong Kong Polytechnic University, Hong Kong (e-mail: nan.guan@polyu.edu.hk).

B. Hu is with the College of Information Science and Technology, Beijing University of Chemical Technology, Beijing 100029, China (e-mail: hubiao@mail.buct.edu.cn).

W. Yi is with the Department of Information Technology, Uppsala University, 75105 Uppsala, Sweden, and also with the School of Computer Science and Engineering, Northeastern University, Shenyang 110819, China (e-mail: yi@it.uu.se).

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the author.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2018.2857359

I. Introduction

INTEGRATING applications with different criticality levels on a shared computing platform has increasingly become a common trend in the design of real-time embedded systems. Such a trend has been observed in the automotive [12] and avionics [17] industries and has led to the emergence of mixed-criticality (MC) systems. An MC task model was proposed by Vestal in his seminal paper [20] about ten years ago, wherein different WCETs are specified for each task on all existing criticality levels, with the one on a higher criticality level being more pessimistic. Since then, many techniques for analyzing and scheduling MC systems have been proposed in the real-time literature (see [7] for a comprehensive review). However, these approaches proposed in nearly a decade still share very impractical assumptions on MC task execution behavior. Specifically, once any high criticality task overruns, the following behaviors are assumed.

1) All low-criticality tasks are abandoned. It is pessimistic to immediately abandon all low-criticality tasks because low-criticality tasks require a certain timing performance as well [12], [19].
2) All high-criticality tasks are assumed to exhibit high criticality behaviors. It is overly pessimistic to bind the mode switches of all high-criticality tasks together in the analysis, as the mode switches of high-criticality tasks are naturally independent.
3) High-criticality tasks are directly transited to the most pessimistic level. This will result in unnecessary resource overbooking because high-criticality tasks rarely reach its most pessimistic WCET estimation during run-time.

A. Related Work

Some solutions have been proposed to partly resolve the above problems. In Table I, we summarize the existing solutions in relation to the three problems described above. These solutions can be broadly categorized into the following classes. The first category of research offers low-criticality tasks a certain degraded service quality when the system is in high-criticality mode. Assumptions of abandoning all low-criticality tasks are relaxed by reducing the dispatch frequency of jobs [19] or by reducing the execution budget of jobs [6], [15]. However, these studies still apply a pessimistic mode-switch strategy.

To address the first and second problems, the second category of studies offer solutions for improving performance for

TABLE I
COMPARISON OF THE EXISTING SOLUTIONS

	P_1 Graceful Degradation	P_2 Independent Mode-Switches	P_3 Multiple-Shot Transition
[19], [6], [15]	√	×	×
[10], [18], [9], [14]	√	√	×
[4], [5]	×	×	√
Our Work	√	√	√

low-criticality tasks by using group-based mode-switch strategies [10], [18]. However, these mode-switch strategies are not flexible enough because the dependencies between low-criticality and high-criticality tasks are statically determined. To relax such dependencies, a new MC model, called flexible MC (FMC) model, was recently proposed in [9], where mode-switches of high-criticality tasks are independent and the service degradation of low-criticality is dynamically updated based on the overruns of high-criticality tasks. Lee *et al.* [14] proposed an MC-ADAPT framework supporting online adaptive task dropping under task-level mode switch that involves using a similar technique. However, the third problem is not addressed in these two state-of-the-art work. In [9] and [14], high-criticality tasks always directly transit to the *most* pessimistic level, in which very pessimistic design parameters are applied.

To support multishot transitions, EDF-VD scheduling algorithm is extended to support a K-level implicit-deadline task system in [4] and [5]. However, the K-level MC task model in [4] and [5] still applies impractical assumptions. Specifically, when the system switches the mode to level k, all the tasks of criticality at least k are assumed to exhibit k-level criticality behaviors (i.e., assumption P_2). All other tasks of criticality less than k are discarded (i.e., assumption P_1).

To the best of our knowledge, no work to date has addressed the above three problems collectively. Compared to existing studies, the motivation of this paper is to find a more fine-grained transition scheme for overrun handling that captures the varying execution behaviors of high-criticality tasks. Instead of always transiting to the *most* pessimistic level, the proposed MC system can undergo intermediate levels to handle overruns with less pessimistic design parameters, such that unnecessary resource over-booking can be avoided. By doing so, a better run-time tradeoff between the penalty of low-criticality tasks and the overruns of high-criticality tasks can be achieved to improve the service quality of low-criticality tasks.

B. Contributions

In this paper, we propose an FMC model with multiple-shot transitions (FMC-MST) operating on a uni-processor platform. Rather than always switching to the *most* pessimistic level (the strategy used in [9] and [14]), the new model allows each high-criticality task to progress over multiple less pessimistic intermediate levels and to scale the deadline nonuniformly on each criticality level. Since high-criticality tasks rarely reach their pessimistic WCET estimations, FMC-MST can avoid unnecessary resource overbooking for overruns by switching high-criticality tasks to less pessimistic intermediate levels. Furthermore, FMC-MST provides a fine-grained transition scheme where mode-switches are independent with these intermediate criticality levels. The overrun of a high-criticality task only raises its own criticality level while others remain at their previous criticality levels. The minimum required low-criticality service degradation is calculated to maintain the balanced system utilization, so as to secure the additional resources requested by a level-transiting task. The contributions of this paper can be summarized as follows.

1) We propose a new EDF-VD-based scheduling for an MC model with multiple-shot transition schemes. Compared to the state-of-the-art work [9], [14], this paper provides a more generalized FMC model that allows high-criticality tasks to progress through multiple criticality levels and to scale deadlines nonuniformly.
2) We develop a run-time schedulability analysis for each independent mode-switch. To improve the service quality of low-criticality tasks, the utilization balance between low-criticality and high-criticality tasks serves as a basic principle for finding an optimal service degradation strategy for low-criticality tasks to compensate for the additional resources requested by multishot overruns of high-criticality tasks.
3) We formally prove the correctness of run-time schedulability analysis for this fine-grained transition scheme.
4) We develop a resource optimization technique that can find resource-efficient level-insertion configurations for FMC-MST task systems under MC timing constraints.

Our evaluation on randomly generated task systems shows that the performance of FMC-MST outperforms the state-of-the-art MC scheduling approaches.

II. BACKGROUND

A. FMC Implicit-Deadline Sporadic Task Model With Multiple Criticality Levels

We consider an MC sporadic task system γ as consisting of a finite collection $\{\tau_1, \tau_2, \ldots, \tau_n\}$ of n MC implicit-deadline sporadic tasks with multiple criticality levels. Each task τ_i in γ generates an infinite sequence of jobs and can be specified by a tuple $\{T_i, \chi_i, \mathcal{C}_i\}$, where:

1) T_i is the minimum job-arrival intervals;
2) χ_i is the total number of criticality levels;
3) $\mathcal{C}_i = (C_i(0), C_i(1), \ldots, C_i(\chi_i - 1))$ is a vector of the worst-case execution times (WCETs). We assume that $C_i(0) \leq C_i(1) \leq \cdots \leq C_i(\chi_i - 1)$.

For the classic dual-criticality system, high-criticality task has two criticality levels with $\chi_i = 2$ and low-criticality task has one criticality level with $\chi_i = 1$. In this paper, we consider an extended dual-criticality task system in which the concepts of high-criticality task and low-criticality task are presented as follows.

Definition 1: In an MC system with multiple criticality levels, tasks with $\chi_i >= 2$ and $\chi_i = 1$ are called high-criticality and low-criticality tasks, respectively.

According to Definition 1, we can divide task set γ into low-criticality task set γ_L and high-criticality task set γ_H. In an MC system with multiple criticality levels, high-criticality tasks are allowed to have several overrun scenarios during run-time. We denote l_i as the criticality level whereby τ_i stays during run-time, and we have $l_i = \{0, 1, 2, \ldots, \chi_i - 1\}$. The mode-switch from level $l_j - 1$ to level l_j can be defined as follows.

Definition 2 (Mode-Switch $M_j^{l_j}$ and $\hat{M}_j^{l_j}$): When high-criticality task τ_j executes for its $C_j(l_j - 1)$ time units without signaling completion, high-criticality task τ_j immediately switches from level $l_j - 1$ to level l_j. This procedure is denoted as mode-switch $M_j^{l_j}$. The closest mode-switch[1] occurring before $M_j^{l_j}$ is denoted as $\hat{M}_j^{l_j}$. For the special case of $l_j = 0$, M_j^0 denotes high-criticality task τ_j executes at level 0.

In FMC-MST, each mode-switch $M_j^{l_j}$ is independent. Mode-switch $M_j^{l_j}$ does not require other high-criticality tasks to exhibit high-criticality behavior. For low-criticality tasks, their execution budget is updated dynamically in accordance with $M_j^{l_j}$. To model the degradation of low-criticality tasks on the point of mode-switch $M_j^{l_j}$, we now introduce the concept of the service level as follows.

Definition 3 (Service Level $z_i(M_j^{l_j})$): When the system has undergone mode switch $M_j^{l_j}$, up to $z_i(M_j^{l_j}) \cdot C_i(0)$ time units can be used for the execution of τ_i in one period T_i.

In this paper, we consider implicit-deadline task systems with task period being equal to the relative deadline (i.e., $T_i = d_i$). The utilization of a task denotes the ratio of its WCET to its period. We define the utilization of task τ_i at level l_i as

$$u_i(l_i) = \frac{C_i(l_i)}{T_i} \quad l_i = \{0, 1, 2, \ldots, \chi_i - 1\}.$$

The total utilization of low-criticality task set in the initial mode (i.e., all high-criticality tasks stay at criticality level 0) is defined as $u_L(0) = \sum_{\tau_i \in \gamma_L} u_i(0)$. According to Definition 3, the degraded utilization of low-criticality tasks on mode-switch $M_j^{l_j}$ can be defined as $u_L(M_j^{l_j}) = \sum_{\tau_i \in \gamma_L} z_i(M_j^{l_j}) \cdot u_i(0)$.

In this paper, we assume that the condition of $z_i(M_j^{l_j}) \leq z_i(\hat{M}_j^{l_j})$ should hold to accommodate the resource overbooking of mode-switch $M_j^{l_j}$. Correspondingly, the system utilization reduction $\Delta u_L(M_j^{l_j})$ of low-criticality tasks on mode-switch $M_j^{l_j}$ can be computed as $u_L(M_j^{l_j}) - u_L(\hat{M}_j^{l_j})$. Since $z_i(M_j^{l_j}) \leq z_i(\hat{M}_j^{l_j})$, we have $\Delta u_L(M_j^{l_j}) \leq 0$.

Remark 1: Note that, in FMC-MST, $\Delta u_L(M_j^{l_j})$ is off-line determined to guarantee a schedulable MC system (see Section III-C). In general, we do not need to specify the settings of $z_i(M_j^{l_j})$ during off-line stage. *Any on-line strategy* on tuning $z_i(M_j^{l_j})$ can be applied as long as it can achieve the required utilization reduction $\Delta u_L(M_j^{l_j})$.

B. EDF-VD Scheduling With Nonuniform Virtual Deadlines

In this paper, we study the schedulability for FMC-MST tasks model under EDF-VD scheduling. The main idea of EDF-VD is to use reduced virtual deadlines to obtain extra slack time for jobs and further decrease the workload of high-criticality tasks after mode-switch.

[1]In general, the closest mode-switch $\hat{M}_j^{l_j}$ before $M_j^{l_j}$ can be any task's prior mode switch.

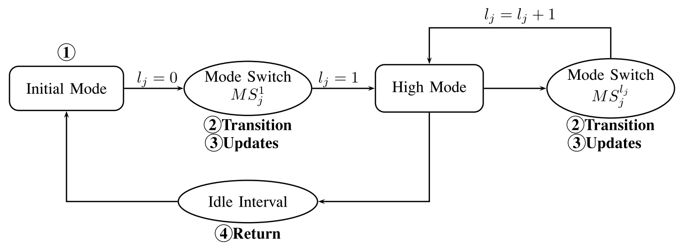

Fig. 1. Execution semantics.

In EDF-VD [3], the virtual deadlines are uniformly scaled by a single deadline scaling factor x and can be defined uniformly by $d_j^v = x \cdot d_j$. In FMC-MST, we allow *non-uniform* deadline scaling factor $x_j^{l_j}$, where $x_j^{l_j} \in (0, 1)$ is a task and criticality level dependent scaling parameter, to nonuniformly set the virtual deadline as $d_j^v(l_j) = x_j^{l_j} \cdot d_j$.

C. Execution Semantics

The execution semantics of a high-criticality task is illustrated in Fig. 1. Compared to the classic MC execution model, FMC-MST model allows independent mode-switches for high-criticality tasks and dynamic service tuning for low-criticality tasks. As shown in Fig. 1, the system initially operates at level 0 (i.e., ①). An overrun of a high-criticality task only triggers itself to shift its criticality level (i.e., ②) and degrades low-criticality service to accommodate this overruns (i.e., ③). A sequence of overruns trigger the system to proceed through multiple criticality levels one by one independently (i.e., ② and ③) until the condition for transiting back is satisfied (i.e., ④). The execution semantics can be summarized as follows.

① *Initial Mode:* All tasks in γ start in level 0 (i.e., $\forall \tau_i, l_i = 0$). As long as no high-criticality task violates its $C_i(0)$, the system remains in level 0. All tasks are scheduled with $C_i(0)$.

② *Transition:* When one job of a high-criticality task τ_j *that is being executed in level* $l_j - 1$ overruns its $C_j(l_j - 1)$ without signaling completion, τ_j only triggers itself to switch into level l_j and update virtual deadline as $d_j^v(l_j)$. However, all other high-criticality tasks still stay in the same criticality level as before.

③ *Updates:* To balance the additional resource demand caused by mode-switch $M_j^{l_j}$, a new service level $z_i(M_j^{l_j})$ is determined and updated to provide degraded service for low-criticality tasks τ_i. At this updating instant, if any low-criticality jobs have completed more than $z_i(M_j^{l_j}) \cdot c_i(0)$ time units of execution, those jobs will be suspended immediately and wait for the budget to be renewed in the next period. Otherwise, low-criticality jobs can continue to use the remaining time budget for their execution.

④ *Return to Low-Criticality Mode:* When the system detects an idle interval [6], the system transits back to the low-criticality mode.

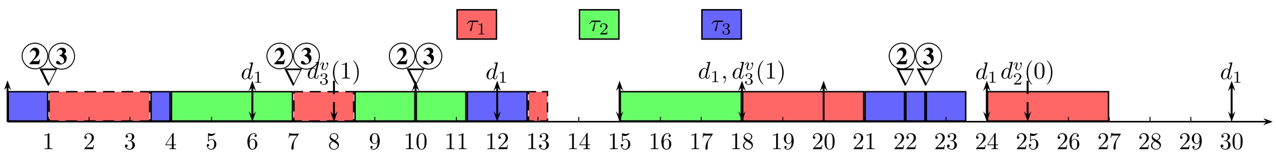

Fig. 2. Illustrative example.

TABLE II
EXAMPLE TASK SET

	χ_i	T_i	$C_i(0)/d_i^v(0)$	$C_i(1)/d_i^v(1)$	$C_i(2)/d_i^v(2)$
τ_1	1	6	3		
τ_2	3	15	$3/10$	$4.5/12.5$	$5.75/15$
τ_3	3	10	$1/5$	$1.5/8$	$2.5/10$

TABLE III
DEGRADED UTILIZATION

Mode Swithch	M_2^1	M_2^2	M_3^1	M_3^2
$\Delta u_L(M_j^{l_j})$	$-\frac{1}{6}$	$-\frac{1}{6}$	$-\frac{1}{12}$	$-\frac{1}{12}$
Budget Reduction	-1	-1	-0.5	-0.5

TABLE IV
TABLE OF NOTATIONS

Symbol	Meaning in the paper
$x_j^{l_j}$	Virtual deadline factor of task τ_j at level l_j
$u_j(l_j)$	Utilization of task τ_j at level l_j
$u_L(M_j^{l_j})$ $(u_L(\hat{M}_j^{l_j}))$	Total utilization of low-criticality tasks after (before) mode switch $M_j^{l_j}$
$\Delta u_L(M_j^{l_j})$	Utilization reduction of low-criticality tasks required to accommodate mode switch $M_j^{l_j}$
γ_H^H	Mode-switched task set $\gamma_H^H = \{\tau_j \in \gamma_H \vert l_j \geq 1\}$
γ_H^L	Non-mode-switched task set $\gamma_H^L = \{\tau_j \in \gamma_H \vert l_j = 0\}$
$a_j^{l_j}$ $(d_j^{l_j})$	Absolute release time (deadline) of the job of high-criticality τ_j that switches to $M_j^{l_j}$

D. Illustrative Example

Now, we give an example to illustrate the related concepts and execution semantics of FMC-MST. Table II gives three tasks, one low-criticality task ($\chi_1 = 1$) and two high-criticality tasks ($\chi_2 = 3$ and $\chi_3 = 3$). For high-criticality tasks, each criticality level l_j ($j = 2, 3$) associates with one virtual deadline $d_j^v(l_j)$, where $l_j \in \{0, 1, 2\}$. Table III gives the required utilization degradation $\Delta u_L(M_j^{l_j})$ for each mode switch to guarantee a schedulable MC system.[2] Fig. 2 depicts the scheduling of MC tasks under execution semantics of FMC-MST, where the symbol ∇ is used to indicate mode-switch occurrence point. In Fig. 2, the jobs are operated under the following rules.

1) Low-criticality task is scheduled with their real deadlines. In Fig. 2, τ_1 is scheduled with $d_1 = 6$.
2) At each mode switch point ∇, operation ② is triggered to update the virtual deadline while operation ③ is triggered to update the execution budget. Now we take the first mode switch as example for illustration. At $t = 1$, the first mode-switch M_3^1 occurs. τ_3 switches its criticality level from $l_3 = 0$ to $l_3 = 1$ with extending virtual deadline as $d_3^v(1) = 8$, while τ_2 stay in the same criticality level as before (i.e., ②). This deadline extension (i.e., $d_3^v(1) = 8$) simultaneously results in the pre-emption of τ_1 at $t = 1$. The execution budgets of low-criticality task τ_1 are decreased from 3 to 2.5 to achieve the required $\Delta u_L(M_3^1)$. τ_1 completes its execution at time instant 3.5 due to using up the budget (i.e., ③).
3) During a busy interval in which multiple overruns occur, the effects of the overruns on budget reduction are independent. For example, during [0, 15], three mode switches ($M_3^1 \triangleright M_2^1 \triangleright M_2^2$) occur sequentially. By Table III, the required budget reduction can be simply calculated as the sum of the one of these three mode switches, that is -2.5. Therefore, τ_1 only has 0.5 time unit for execution.

[2]The derivations for determining $\Delta u_L(M_j^{l_j})$ are illustrated in Ex 1.

III. SCHEDULABILITY ANALYSIS AND RESOURCE OPTIMIZATION

Our FMC-MST model is a more generalized model that allows multiple less pessimistic criticality levels and nonuniform deadline scaling. In this section, we present a utilization-based schedulability analysis for FMC-MST scheduling algorithm. We first analyze online schedulability for a single mode switch $M_j^{l_j}$, by which the minimum low-criticality service degradation can be derived to accommodate the resource overbooking of a mode switch. In Section III-A, we provide a high-level overview for this online schedulability analysis and attempt to communicate the intuition behind the algorithm design by means of an example. We then provide a more comprehensive description in Section III-B to prove the correctness of Theorem 1. In Section III-C, we check whether a task set is schedulable by FMC-MST under arbitrary sequences of mode switches. In Section III-D, we develop an intermediate level insertion technology and attempt to solve the problem of how to determine intermediate levels for high-criticality tasks to minimize the penalties of low-criticality tasks without sacrificing MC schedulability. We finally prove some important properties of FMC-MST. Table IV shows the notation used throughout this paper.

A. Sufficient Schedulability Test on Transition Case $M_j^{l_j}$

In this section, we provide a high-level overview of online schedulability analysis for one mode switch, and introduce the derived schedulability test condition in Theorem 1. With these conditions, we can adaptively determine how much of execution budget can be reserved for low-criticality tasks to handle each intermediate overrun while ensuring a schedulable system during run-time. Without loss of generality, we consider a general transition case $M_j^{l_j}$ where high-criticality task τ_j switches from level l_j-1 to l_j, and assume the system is MC-schedulable on level l_j-1. To accommodate $M_j^{l_j}$, the minimum required utilization reduction $\Delta u_L(M_j^{l_j})$ can be determined by Theorem 1.

Theorem 1: For mode-switch $M_j^{l_j}$ with $l_i \geq 1$, when high-criticality task τ_j overruns its $C_j(l_j-1)$, the system is schedulable when the following conditions are satisfied:

$$\Delta u_L\left(M_j^{l_j}\right) + \frac{u_j(l_j)}{x_j^{l_j}} - \frac{u_j(l_j-1)}{x_j^{l_j-1}} \leq 0 \tag{1}$$

$$\Delta u_L\left(M_j^{l_j}\right) + \frac{u_j(l_j) - u_j(l_j-1) + p_j(l_j)}{1 - x_j^{l_j-1}} \leq 0 \tag{2}$$

$$\Delta u_L\left(M_j^{l_j}\right) \leq 0 \tag{3}$$

$$\frac{u_j(l_j)}{x_j^{l_j}} \leq u_j(\chi_j - 1) \tag{4}$$

where $p_j(l_j)$ are constrained by

$$p_j(l_j) \leq 0 \tag{5}$$

$$\sum_{l_j=1}^{\chi_j-1} p_j(l_j) = u_j(0) - \frac{u_j(0)}{x_j^0} \tag{6}$$

with the initial utilization condition on criticality level 0

$$u_L(0) + \sum_{\tau_j \in \gamma_H} \frac{u_j(0)}{x_j^0} \leq 1 \tag{7}$$

$$\frac{u_j(0)}{x_j^0} \leq u_j(\chi_j - 1). \tag{8}$$

Intuition: The intuition behind Theorem 1 is to maintain balanced system utilization during the transitions. The conditions can be explained as follows. Equation (7) ensures MC schedulability when the system stays in initial mode [3]. An event of overrun of high-criticality task normally results in an increase in virtual and overrun utilization due to resource overbooking. By analyzing the difference in virtual and overrun utilization, (1) and (2) serve as an efficient way to maintain the resource balance between the penalty of low-criticality tasks and the overruns of high-criticality tasks. Via (1) and (2), the minimum required utilization reduction $\Delta u_L(M_j^{l_j})$ can be determined to maintain the balanced system utilization, so as to secure the additional resources requested by a level-transiting task. According to [14], high-criticality task with $(u_j(l_j)/x_j^{l_j}) \geq u_j(\chi_j - 1)$ will produce schedulability loss. Therefore, additional constraints (4), (8) are imposed to avoid the performance loss during transitions. In order to provide an intuition of how the proposed analysis works, we apply Theorem 1 on a simple task set and calculate the required utilization degradation for guaranteeing MC schedulability of a single mode switch.

TABLE V
FEASIBLE SETTINGS

Mode Swithch	M_2^1	M_2^2	M_3^1	M_3^2
$p_j(l_j)$	$-\frac{2}{45}$	$-\frac{1}{18}$	$-\frac{1}{60}$	$-\frac{1}{12}$
$x_j^{l_j-1}$	$\frac{2}{3}$	$\frac{5}{6}$	$\frac{1}{2}$	$\frac{4}{5}$

Example 1: Consider a task set in Table II. Feasible settings[3] on $p_j(l_j)$ and $x_j^{l_j}$ are listed in Table V, so that conditions (4)–(8) are satisfied. In the following, we take the mode switch M_2^1 as an example to illustrate the derivation process of the required utilization degradation $\Delta u_L(M_2^1)$. According to (1)–(3) in Theorem 1, utilization degradation $\Delta u_L(M_2^1)$ should satisfy the following conditions to accommodate a feasible mode switch M_2^1:

$$\begin{aligned}
&\Delta u_L\left(M_2^1\right) \\
&\leq \min\left(\underbrace{-\left(\frac{u_2(1)}{x_2^1} - \frac{u_2(0)}{x_2^0}\right)}_{\text{virtual utilization}}, \underbrace{-\frac{u_2(1) - u_2(0) + p_2(1)}{1 - x_2^0}}_{\text{overrun utilization}}, 0\right) \\
&= -\frac{1}{6}.
\end{aligned}$$

The similar derivation can be operated to obtain utilization degradation for other mode switches, as presented in Table III.

B. Proof of the Correctness

We now prove the correctness of the schedulability test condition presented in Theorem 1. The proof process involves three steps. We first determine the initial conditions to ensure the schedulability of tasks in initial mode (7) and to satisfy the necessary boundary constraints (3), (4), and (8). In the second step, we prove the correctness of the sufficient condition [i.e., (1)] to ensure MC schedulability after mode switch $M_j^{l_j}$. In the third step, we propose a sufficient schedulability condition [i.e., (2)] to maintain balanced overrun utilization as the system undergoes mode transition $M_j^{l_j}$.

1) Initial Conditions: The basic assumption $z_i(M_j^{l_j}) \leq z_i(\hat{M}_j^{l_j})$ implies (3). According to [3], we can use (7) to ensure MC schedulability of in level 0. Equations (4) and (8) restrict resource utilization to levels less than those achieved in the most pessimistic level (i.e., level $\chi_j - 1$). Otherwise, tasks can directly execute in level $\chi_j - 1$ for efficient resource use [14].

2) Virtual Utilization Balance Equation: We now show how to ensure MC schedulability after mode switch $M_j^{l_j}$ occurs. This is achieved via virtual utilization balance analysis before and after mode switch $M_j^{l_j}$. By replacing the period as virtual deadline, virtual utilization of each high-criticality task τ_j on level l_j is computed as $(u_j(l_j)/x_j^{l_j})$. $u_\gamma^v(\hat{M}_j^{l_j})$ and $u_\gamma^v(M_j^{l_j})$ denote the virtual utilization of task set γ before and after mode switch $M_j^{l_j}$, respectively. To ensure the correctness of system

[3]Feasible settings can be off-line determined by formulated CSP problem presented in Section III-C.

behaviors after mode switch $M_j^{l_j}$, system virtual utilization $u_\gamma^v(M_j^{l_j})$ must meet the following condition:

$$u_\gamma^v\left(M_j^{l_j}\right) = u_L\left(M_j^{l_j}\right) + \sum_{\tau_j \in \gamma_H} \frac{u_j\left(l_j\right)}{x_j^{l_j}} \leq 1. \tag{9}$$

After mode switch $M_j^{l_j}$, high-criticality task τ_j overruns $C_j(l_j - 1)$ and shifts from level $l_j - 1$ to level l_j. With the exception of high-criticality task τ_j, all other high-criticality tasks remain at their respective criticality levels without changing the utilization. Therefore, an increase in the virtual utilization of high-criticality tasks can be determined as $(u_j(l_j)/x^{l_j}) - ([u_j(l_j - 1)]/[x_j^{l_j - 1}])$. For low-criticality tasks, low-criticality utilization is degraded from $u_L(\hat{M}_j^{l_j})$ to $u_L(M_j^{l_j})$ due to resource overbooking of overruns. Therefore, the difference in system virtual utilization can be formulated as

$$\begin{aligned} & u_\gamma^v\left(M_j^{l_j}\right) - u_\gamma^v\left(\hat{M}_j^{l_j}\right) \\ &= \underbrace{u_L\left(M_j^{l_j}\right) - u_L\left(\hat{M}_j^{l_j}\right)}_{\text{Utilization Reduction}} + \underbrace{\frac{u_j\left(l_j\right)}{x_j^{l_j}} - \frac{u_j\left(l_j - 1\right)}{x_j^{l_j - 1}}}_{\text{Utilization Increment}} \\ &= \underbrace{\Delta u_L\left(M_j^{l_j}\right) + \frac{u_j\left(l_j\right)}{x_j^{l_j}} - \frac{u_j\left(l_j - 1\right)}{x_j^{l_j - 1}}}_{(1)}. \end{aligned} \tag{10}$$

As the system is schedulable before mode switch $M_j^{l_j}$, we have $u_\gamma^v(\hat{M}_j^{l_j}) \leq 1$. Hence, we find that (1) ensures the correctness of $u_\gamma^v(M_j^{l_j}) \leq u_\gamma^v(\hat{M}_j^{l_j}) \leq 1$ to guarantee MC schedulability after the mode-switch.

3) Overrun Utilization Balance Equation: As the third step, we prove that the condition presented in (2) is sufficient to ensure the MC schedulability during the transition phase. We adopt the similar proof strategy based on [4] and [9] and prove it by contradiction. Suppose that there is a time interval $[0, t_f]$ such that the system undergoes mode switch $M_j^{l_j}$ and the first deadline miss occurs at t_f. Let J denote the minimal set[4] of jobs released from task set γ for which a deadline is missed. $\eta_i^{l_j}(t_1, t_2)$ denotes cumulative execution time of task τ_i when the system undergoes the mode-switch $M_j^{l_j}$ during the interval $(t_1, t_2]$. $N_\gamma^{l_j}$ denotes the sum of $\eta_i^{l_j}(0, t_f)$ for all tasks in γ. Since the first deadline miss occurs at t_f, we have $N_\gamma^{l_j} > t_f$. In the following, we will show the upper bound of $N_\gamma^{l_j}$ is less than t_f, which leads to a contradiction.

To calculate the upper bound of $N_\gamma^{l_j}$, we start the proof by introducing auxiliary lemmas to analyze the upper bound of cumulative execution time for high-criticality tasks (i.e., Lemmas 1 and 2) and low-criticality tasks (i.e., Lemma 3).

High Criticality Tasks: Since the mode switches are independent, high-criticality tasks can be divided into mode-switched task set γ_H^H and nonmode-switched task set γ_H^L. Now, we derive upper bounds of the cumulative execution time for both types of high-criticality tasks.

Lemma 1: For high-criticality task τ_j of task set γ_H^H, the cumulative execution time $\eta_j^{l_j}(0, t_f)$ can be bounded by

$$\frac{a_j^1}{x_j^0} \cdot u_j(0) + \left(t_f - a_j^1\right) u_j(1) + \sum_{r_j=2}^{l_j} \left(t_f - a_j^{r_j}\right) \Delta u_j(r_j) \tag{11}$$

where $\Delta u_j(r_j) = u_j(r_j) - u_j(r_j - 1)$.

Proof: Recall that $a_j^{r_j}$ is the absolute release time of the job executed on level r_j. High-criticality task τ_j progresses though l_j levels. Therefore, the analysis duration can be divided into $l_j + 1$ time segments, as shown in Fig. 3. During time segment $[a_j^{r_j}, a_j^{r_j+1}]$, the execution requirement per job is bounded by $c_j(r_j)$. For ease of presentation, we use $a_j^{l_j+1} = t_f$. Considering l_j time segments shown in Fig. 3, the cumulative execution time $\eta_j^{l_j}(0, t_f)$ can be bounded as

$$\begin{aligned} \eta_j^{l_j}(0, t_f) &\leq a_j^1 \cdot u_j(0) + \sum_{r_j=1}^{l_j} \left(a_j^{r_j+1} - a_j^{r_j}\right) \cdot u_j\left(r_j\right) \\ &\leq \frac{a_j^1}{x_j^0} u_j(0) + \left(a_j^2 - a_j^1\right) u_j(1) \\ &\quad + \underline{\sum_{r_j=2}^{l_j} \left(a_j^{r_j+1} - a_j^{r_j}\right) \cdot u_j(r_j)}. \end{aligned} \tag{12}$$

Since $u_j(r_j) = \sum_{k=2}^{r_j}(u_j(k) - u_j(k-1)) + u_j(1)$, we have

$$\begin{aligned} & \sum_{r_j=2}^{l_j} \left(a_j^{r_j+1} - a_j^{r_j}\right) \cdot u_j\left(r_j\right) \\ &= \left(a_j^{r_j+1} - a_j^2\right) u_j(1) + \sum_{r_j=2}^{l_j} \sum_{k=2}^{r_j} \left(a_j^{r_j+1} - a_j^{r_j}\right) \\ &\quad \times \left(u_j(k) - u_j(k-1)\right) \\ &= \left(a_j^{r_j+1} - a_j^2\right) u_j(1) + \sum_{k=2}^{l_j} \sum_{r_j=k}^{l_j} \left(a_j^{r_j+1} - a_j^{r_j}\right) \\ &\quad \times \left(u_j(k) - u_j(k-1)\right) \\ &= \left(a_j^{r_j+1} - a_j^2\right) u_j(1) + \sum_{k=2}^{l_j} \left(a_j^{l_j+1} - a_j^k\right) \left(u_j(k) - u_j(k-1)\right). \end{aligned} \tag{13}$$

Substituting the marked item in (12) with (13), $\eta_j^{l_j}(0, t_f)$ can be reformulated as

$$\frac{a_j^1}{x_j^0} u_j(0) + \left(a_j^{l_j+1} - a_j^1\right) u_j(1) + \sum_{k=2}^{l_j} \left(a_j^{l_j+1} - a_j^k\right) \left(u_j(k) - u_j(k-1)\right). \tag{14}$$

Therefore, $\eta_j^{l_j}(0, t_f)$ can be bounded as (11) by replacing $a_j^{l_j+1}$ and k with t_f and r_j, respectively. ■

[4]This minimality means that if any job is removed from J, the remainder of J will be schedulable.

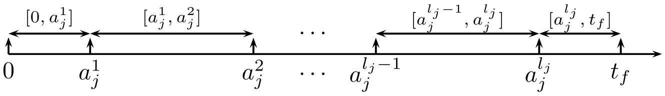

Fig. 3. Time segments.

Lemma 2 (From [9]): High-criticality task τ_j in task set γ_H^L has

$$\eta_j^0\left(0, t_f\right) \leq \frac{t_f}{x_j^0} u_j(0). \tag{15}$$

Low Criticality Tasks: We now derive an upper bound on the cumulative execution time $\eta_i^{l_j}(0, t_f)$ for low-criticality tasks using a proof strategy similar to that used in [9].

Lemma 3: For low-criticality task τ_i, the cumulative execution time $\eta_i^{l_j}(0, t_f)$ can be upper bounded by

$$t_f \cdot u_i(0) + \sum_{\tau_j \in \gamma_H^H} \sum_{r_j=1}^{l_j} \psi_i^{r_j} \tag{16}$$

with *difference term* $\psi_i^{r_j} = (t_f - a_j^{r_j})(1 - x_j^{r_j-1})\Delta u_i(M_j^{r_j})$.

Proof: We will only sketch the proof here as it is similar to the proof in [9]. The detailed proof is presented in Appendix A in the supplementary material. Following the proof strategy in [9], we analyze the difference of the cumulative execution time before and after mode-switch $M_j^{l_j}$ and prove that the difference can be uniformly upper bounded by *difference term* $\psi_i^{l_j}$. By visiting all mode switches $M_j^{r_j}$, the upper bound of $\eta_i^{l_j}(0, t_f)$ can be obtained. ■

Total Cumulative Requirements: Now, we sum the cumulative requirements over all tasks given as (17) and prove the sufficient condition (2). The complete derivation of $N_\gamma^{l_j}$ is given in Appendix B in the supplementary material

$$\begin{aligned}
N_\gamma^{l_j} &= \sum_{\tau_i \in \gamma_L} \eta_i^{l_j}\left(0, t_f\right) + \sum_{\tau_j \in \gamma_H^L} \eta_j^0\left(0, t_f\right) + \sum_{\tau_j \in \gamma_H^H} \eta_j^{l_j}\left(0, t_f\right) \\
&\leq t_f + \sum_{\tau_j \in \gamma_H^H} \left(t_f - a_j^1\right) \\
&\quad \times \left(\left(1 - x_j^0\right)\Delta u_L\left(M_j^1\right) + \Delta u_j(1) + \underbrace{u_j(0) - \frac{u_j(0)}{x_j^0}}_{(6)} \right) \\
&\quad + \sum_{\tau_j \in \gamma_H^H} \sum_{r_j=2}^{l_j} \left(t_f - a_j^{r_j}\right)\left(\left(1 - x_j^{r_j-1}\right)\Delta u_L\left(M_j^{r_j}\right) + \Delta u_j\left(r_j\right)\right) \\
&= t_f + \sum_{\tau_j \in \gamma_H^H} \left(t_f - a_j^1\right)\left(\left(1 - x_j^0\right)\Delta u_L\left(M_j^1\right) + \Delta u_j(1) \right. \\
&\qquad \left. + \sum_{l_j=1}^{\chi_j-1} p_j\left(l_j\right) \right) \\
&\quad + \sum_{\tau_j \in \gamma_H^H} \sum_{r_j=2}^{l_j} \left(t_f - a_j^{r_j}\right)\left(\left(1 - x_j^{r_j-1}\right)\Delta u_L\left(M_j^{r_j}\right) + \Delta u_j\left(r_j\right)\right)
\end{aligned}$$

Since $a_j^1 \leq a_j^2 \leq \cdots \leq a_j^{l_j} < t_f$ and $p_j(r_j) \leq 0$

$$\begin{aligned}
&\leq t_f + \sum_{\tau_j \in \gamma_H^H} \sum_{r_j=1}^{l_j} \left(t_f - a_j^{r_j}\right)\left(\left(1 - x_j^{r_j-1}\right)\Delta u_L\left(M_j^{r_j}\right) \right. \\
&\qquad \left. + \Delta u_j\left(r_j\right) + p_j\left(r_j\right)\right).
\end{aligned} \tag{17}$$

The assumed deadline miss implies $N_\gamma > t_f$. That is,

$$\begin{aligned}
&\sum_{\tau_j \in \gamma_H^H} \sum_{r_j=1}^{l_j} \left(t_f - a_j^{r_j}\right)\left(\left(1 - x_j^{r_j-1}\right)\Delta u_L\left(M_j^{r_j}\right) \right. \\
&\qquad \left. + \Delta u_j\left(r_j\right) + p_j\left(r_j\right)\right) > 0.
\end{aligned}$$

Taking the contrapositive, we have

$$\begin{aligned}
&\sum_{\tau_j \in \gamma_H^H} \sum_{r_j=1}^{l_j} \left(t_f - a_j^{r_j}\right) \\
&\times \left(\underbrace{\left(1 - x_j^{r_j-1}\right)\Delta u_L\left(M_j^{r_j}\right) + \Delta u_j\left(r_j\right) + p_j\left(r_j\right)}_{(2)} \right) \leq 0.
\end{aligned} \tag{18}$$

Since $t_f - a_j^{r_j} > 0$, it is sufficient to ensure the system schedulability of task set γ by guaranteeing (2) holds for each mode switch $M_j^{r_j}$. In (18), the constraints imposed on each mode switch $M_j^{r_j}$ are consistent to each other. Based on this property, the constraints imposed on current mode switch $M_j^{l_j}$ imply the condition (2), guaranteeing MC schedulability during the transition phase.

C. Feasibility of Algorithm

Theorem 1 gives an online schedulability test condition *only* for a single transition. It is yet unclear how to off-line determine whether a task set is schedulable by FMC-MST under arbitrary sequences of mode switches. In this section, we present the off-line schedulability test conditions for a task set with specified criticality levels. To guarantee schedulability, we must ensure that FMC-MST can successfully schedule the task set under any execution scenario during run-time. Therefore, to show that the task set is MC-schedulable, we need to satisfy the following two conditions.

Condition A: We need to guarantee the feasibility of each mode-switch. Therefore, constraints (1)–(8) for each mode-switch must be satisfied.

Condition B: We must ensure the system-wide feasibility. As shown in Theorem 1, each overrun will result in a decreased low-criticality utilization. For low-criticality tasks, we must show remaining low-criticality utilization should not fall below a level of 0 under the worst-case overrun scenario, that is each high-criticality task τ_j reaches criticality level $\chi_j - 1$. Therefore, we require

$$\sum_{\tau_j \in \gamma_H} \sum_{l_j=1}^{\chi_j-1} \Delta u_L\left(M_j^{l_j}\right) + u_L(0) \geq 0. \tag{19}$$

For condition A, constraints (1)–(5) must be subjected to all mode switches with $\forall \tau_j \in \gamma_H$ and $l_j = 1, \ldots, \chi_j - 1$, while constraint (6) should be subjected to all high tasks with $\forall \tau_j \in \gamma_H$. By combining all of these conditions, we can formulate the offline schedulability problem as a constraint satisfaction problem (CSP). Any insertion solution of intermediate levels whose states satisfy a number of constraints in the derived CSP problem can guarantee a feasible scheduling system. We use the following example to illustrate how to evaluate the schedulability of the given insertion solution.

Example 2: Consider the example task set with the dedicated insertion solution given in Table II and the settings listed in Table V. We have already demonstrated condition A is satisfied, as illustrated in Example 1. For condition B, we know it is also satisfied by simply checking

$$\sum_{\tau_j \in \gamma_H} \sum_{l_j=1}^{2} \Delta u_L\Big(M_j^{l_j}\Big) + u_L(0) = -\frac{1}{6} - \frac{1}{6} - \frac{1}{12} - \frac{1}{12} + \frac{3}{6} = 0.$$

Therefore, the example task set in Table II is MC-schedulable.

D. Resource Optimization

Above, we prove a metric for evaluating the schedulability of an MC task set with specified level-insertion configurations. However, for an MC task set with two bounded criticality levels [i.e., $C_j(0)$ and $C_j(\chi_i - 1)$ are known], how to specify a reasonable level-insertion configuration for each high-criticality task is still not known yet. In this section, we will study the off-line resource optimization problem (ROP) with the aim of finding the resource-efficient level-insertion configuration for the FMC-MST task system within MC timing constraints.

In general, the probability that the execution time of high-criticality task reaches its most pessimistic WCET estimation is quite low. However, in EDF-VD scheduling, high-criticality tasks always transit from low-criticality level to the *most* pessimistic level once an overrun occurs. To avoid unnecessary resource over-booking, we can insert several intermediate levels to handle the less pessimistic overruns. The intermediate level to take depends on the real execution time of high-criticality tasks. In this paper, we use the distribution of the execution time of high-criticality task τ_j to compute the probability of overruns. The cumulative distribution function $F_j(t)$ is used to model the diversity of execution time of high-criticality task τ_j during run-time. Hence, the probability of the overrun $M_j^{l_j}$ that the execution time of high-criticality task τ_j falls in $[c_j(l_j), c_j(l_j + 1)]$ can be represented as $F_j(c_j(l_j + 1)) - F_j(c_j(l_j))$. When high-criticality task reaches criticality level l_j, the utilization of low-criticality tasks require to decrease $-\sum_{r_j=0}^{l_j} \Delta u_L(M_j^{r_j})$. In the off-line stage, we introduce a QoS function (20) with the aim to minimize the average low-criticality utilization decrease. Based on this objective and the aforementioned constraints, the ROP is formulated as:

$$\textbf{ROP: min } -\sum_{\tau_j \in \gamma_H} \sum_{l_j=1}^{\chi_j - 1} \Big(F_j\big(c_j(l_j+1)\big) - F_j\big(c_j(l_j)\big)\Big)$$

$$\sum_{r_j=0}^{l_j} \Delta u_L\Big(M_j^{r_j}\Big)$$

$$\text{s.t.} \begin{cases} \textbf{ConditionA}: & \text{Equation (1)} - (8) \\ & \text{for all mode switches} \\ \textbf{ConditionB}: & \text{Equation (19).} \end{cases} \quad (20)$$

The objective function shown above is subjected to the constraints listed in the CSP formulation (conditions A and B). Given an MC task set where two bounded execution times $[C_j(0), c_j(\chi_j - 1)]$ are specified for each high-criticality task, the resource optimization formulation can automatically generate a feasible level-insertion configuration with intermediate execution time $c_j(l_j)$ and deadline scaling factor $x_j^{l_j}$ for each high-criticality task.

Complexity: Due to nonlinear items in the constraints, the ROP (20) is a nonlinear optimization problem (NLP). For a task set with M high-criticality tasks and L criticality levels, then NLP problem has $4M(L-1) + M + 2$ constraints and $4M(L-2) + 3$ real variables. Hence, the number of variables and constraints is polynomially bounded to the size of the input problem, and it can be solved by a polynomial-time heuristic [11].

Properties: We now provide important properties to show the efficiency of FMC-MST.

Property 1: Criticality level insertions operated by ROP do not degrade the schedulability of FMC-MST.

Proof: We consider a general case that a task set is MC-schedulable by FMC-MST with L criticality levels. ROP formulation generates level-insertion configuration $[\Delta u_L(M_j^{l_j}), u_j(l_j), x_j^{l_j}, p_j(l_j)]$ for each criticality level l_j of high-criticality task τ_j. In general, without changing the previous configurations of L levels, one can insert $L+1$th level with the following configuration:

$$\Delta u_L\Big(M_j^{l_j+1}\Big) = 0, u_j\big(l_j+1\big) = u_j\big(l_j\big), x_j^{l_j+1} = x_j^{l_j}$$
$$p_j\big(l_j+1\big) = 0. \quad (21)$$

The new configuration still satisfies the CSP. Therefore, the task set is still MC-schedulable. ■

Property 2: FMC-MST with two criticality levels dominates EDF-AD-E [14] in terms of MC-schedulability.

Proof: For FMC-MST with two criticality levels (i.e., $\chi_j = 2$), $u_j(0)$ and $u_j(1)$ are equivalent to low-criticality and high-criticality utilization in EDF-AD-E, respectively. For task set γ, the high-criticality task set γ_H can be divided into HI-mode-preferred task set $\gamma_H^F = \{\tau_j \in \gamma_{HI} | (u_j(0)/x_j^0) \geq u_j(1)\}$ and non-HI-mode-preferred task set $\gamma_H - \gamma_H^F$, respectively. Assume task set γ is MC-schedulable by EDF-AD-E [14]. Therefore, the following conditions must be satisfied to ensure

MC schedulability according to [14]

$$u_L(0) + \min_{\tau_j \in \gamma_H} \left(\frac{u_j(0)}{x}, u_j(1) \right) \le 1 \tag{22}$$

$$x \cdot u_L(0) + u_H(1) \le 1. \tag{23}$$

In general, we can always find a lower-bound factor $\hat{x}$ that satisfies $u_L(0) + \min_{\tau_j \in \gamma_{HI}}([u_j(0)/\hat{x}], u_j(1)) = 1$ and (23).

To achieve equivalent behavior, we assign $x_j^0 = (u_j(0)/u_j(1))$ for HI-mode-preferred tasks and $\hat{x}$ for non-HI-mode-preferred tasks when applying FMC-MST. By this equivalence transformation, we can make the following observations for the CSP formulation.

1) Equations (7) and (22) are equivalent.
2) $\Delta u_L(M_j^0) = 0$ holds for HI-mode-preferred tasks.
3) For non-HI-mode-preferred tasks, the constraints (1)–(6) can be equivalently merged as (2).

Based on above observations, by (2) and (19), one can derive (23) and guarantee a feasible CSP problem for FMC-MST

$$-u_L(0) \le \sum_{\tau_j \in \gamma_H} \Delta u_L\left(M_j^1\right) \le \frac{\sum_{\tau_j \in \gamma_H - \gamma_H^F} \left(\frac{u_j(0)}{\hat{x}} - u_j(1) \right)}{1 - \hat{x}}$$

$$\rightleftharpoons -u_L(0) \le \frac{\sum_{\tau_j \in \gamma_H - \gamma_H^F} \left(\frac{u_j(0)}{\hat{x}} - u_j(1) \right)}{1 - \hat{x}}$$

$$\rightleftharpoons \hat{x} \cdot u_L(0) + u_H(1) \le u_L(0) + \sum_{\tau_j \in \gamma_H - \gamma_H^F} \frac{u_j(0)}{\hat{x}}$$

$$+ \sum_{\tau_j \in \gamma_H^F} u_j(1). \tag{24}$$

From the definition of γ_H^F and $\gamma_H - \gamma_H^F$

$$\rightleftharpoons \hat{x} \cdot u_L(0) + u_H(1) \le u_L(0) + \min_{\tau_j \in \gamma_H} \left(\frac{u_j(0)}{\hat{x}}, u_j(1) \right).$$

From the definition of $\hat{x}$

$$\rightleftharpoons \hat{x} \cdot u_L(0) + u_H(1) \le 1.$$

Therefore, we can conclude when any task set γ is MC-schedulable by EDF-AD-E [14], it is also MC-schedulable by FMC-MST with two criticality levels. ■

Property 3: FMC-MST with L criticality levels inserted by ROP dominates EDF-AD-E [14] in terms of MC-schedulability.

Proof: This can be directly proved by Properties 1 and 2. ■

IV. Evaluation

A. Experiment Setup

In this section, we conduct the simulation experiments to evaluate the effectiveness of FMC-MST by an extensive comparison to state-of-the-art approaches: EDF-AD-E [14], FMC [9], IMC [15], EDF-VD [3]. Our experiments were conducted based on randomly generated MC task systems. We adopt the same workload generation algorithm as that used in [3], [8], and [10] to randomly generate task sets with two criticality levels. In FMC-MCL, two criticality levels act as the lowest and highest criticality levels (i.e., $l_j = 0$ and $l_j = \chi_j - 1$). The resource optimization approach presented in Section III-D will automatically insert the intermediate levels between these two levels. For ease of presentation, we denote these two criticality levels as LO and HI levels during the generation process. In particular, the various parameters[5] of each task are generated in the following ways.

1) For each task τ_i, low-criticality utilization u_i^{LO} is a real number drawn at random from $[0.05, 0.15]$.[6]
2) R_i denotes the ratio of u_i^{HI}/u_i^{LO} for every high-criticality task, which is a real number drawn uniformly at random from [1, 5].
3) Task period T_i of each task is an integer drawn uniformly at random from [100, 1000].
4) pCri denotes the probability that a task τ_i is a high-criticality task, and we set it as 0.5. When τ_i is a low-criticality task, then set $C_i^{LO} = \lfloor u_i^{LO} \cdot T_i \rfloor$. Otherwise, set $C_i^{LO} = \lfloor u_i^{LO} \cdot T_i \rfloor$ and $C_i^{HI} = \lfloor u_i^{LO} \cdot R_i \cdot T_i \rfloor$.[7]

One task is generated at a time until $u_B - 0.05 \le \max\{u_{LO}^{LO} + u_{HI}^{LO}, u_{HI}^{HI}\} \le u_B$.

As stated in Remark 1, FMC-MST provides a generalized degradation strategy. For the evaluation, we adopt dropping-off strategy where low-criticality tasks are partly dropped by assigning $z_i(M_j^{l_j}) = 0$ for dropped tasks. We quantitatively compare FMC-MST with above state-of-the-art approaches in terms of offline schedulability and online performance. Following [9] and [10], online low-criticality performance is measured by the percentage of finished LC jobs (PFJ). PFJ defines the ratio of the number of finished jobs of LO-critical tasks over the total number of jobs released in a given time interval. During the simulation, the execution distribution in [16], which is a straight line on $[C_i(0), C_i(\chi_i - 1)]$ with probabilities given on a log scale, is used to generate the overrun execution time for jobs of high-criticality tasks. The system takes the intermediate level according to the actual execution time. To ensure fair comparisons, we generate a job trace for each generated task set in off-line and use this unified job trace to obtain the PFJ for all compared schemes during run-time.

B. Results

We first demonstrate the effectiveness of FMC-MST compared with state-of-the-art approaches: FMC [9], EDF-AD-E [14], IMC [15], and EDF-VD [3], in which high-criticality tasks always directly enter the most pessimistic execution mode once overrun occurs. We vary utilization bounds u_B from 0.7 to 0.95 with step size of 0.05, to evaluate offline schedulability and online performance. For FMC-MST, each high-criticality task are inserted with three intermediate levels. Each data-point was obtained by randomly generating 1000 task sets. Fig. 4 shows the acceptance ratio and average PFJ for the compared approaches. The left-axis shows PFJ values achieved for low-criticality tasks represented by the bar

[5]We also follow [13] to evaluate the performance under different settings. More results are available online [1].

[6]In FMC-MCL, u_i^{LO} and u_i^{HI} correspond to $u_i(0)$ and $u_i(\chi_j - 1)$, respectively.

[7]In FMC-MCL, C_i^{LO} and C_i^{HI} correspond to $C_i(0)$ and $C_i(\chi_j - 1)$, respectively.

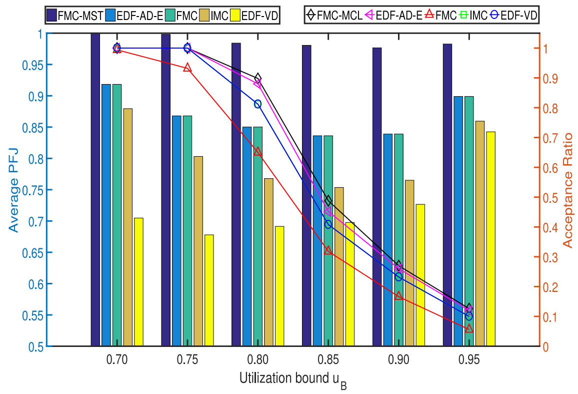

Fig. 4. Performance with varying utilization bound.

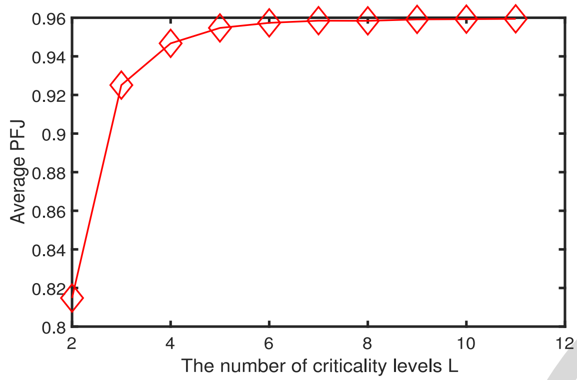

Fig. 5. Impact of the number of criticality levels L.

graphs, and the right-axis shows acceptance ratios represented by line graphs.

As shown in Fig. 4, FMC-MST can provide more low-criticality service without sacrifice in the MC schedulability. We can observe the following trends.

1) FMC-MST outperforms all the compared approaches in terms of support for low-criticality execution. This is expected because one-shot transition scheme-based approaches always switch to the level with applying the most pessimistic design parameters. In contrast, FMC-MST can capture the varying execution behavior of high-criticality tasks and can penalize low-criticality tasks more precisely according to the overrun demands of high-criticality tasks.
2) FMC-MST dominates all the EDF-VD-based scheduling algorithms with one-shot transition scheme. This schedulability performance gain is attributed to the fact that FMC-MST provides a generalized MC model where a nonuniform deadline scaling is a relaxation of EDF-VD-based schedulings [9], [14] and might cause more task sets to be deemed schedulable.

Next, we will show how the number of intermediate levels L will impact the effectiveness of FMC-MST. In this experiment, varying L from 2 to 11, we conduct the simulation on random MC task sets with $u_B = 0.85$. Fig. 5 shows online low-criticality performance under different settings on L. As shown in Fig. 5, the average PFJ increases with the number of insertion levels L. The reason for this trend is that the more insertion levels generally imply more opportunities for handling the less pessimistic overruns during run-time, which can avoid overbooking unnecessary resources.

We finally evaluate the computation time for deriving automatic intermediate level insertion by solving the formulated optimization problem presented in Section III. According to the parameters of task sets presented above, we can automatically generate an optimization problem and use the APMonitor optimization suite [2] to solve it. For all task set tested above, the selected optimization tool can generate results within 8.5 s. The results show that the formulated optimization problem can be solved efficiently.

V. Conclusion

We present a generalized FMC model that enables independent multiple-shot transitions for high-criticality tasks. A run-time schedulability test condition is successfully derived, which serves as a basis principle to find an optimal service degradation strategy for low-criticality tasks. We develop a resource optimization formulation to maximize the run-time low-criticality service quality without sacrificing MC schedulability. Experimental results illustrate the efficiency of the proposed approach.

References

[1] G. Chen *et al.* (2018). *EDF-VD Scheduling of Flexible Mixed-Criticality System With Multiple-Shot Transitions*. [Online]. Available: https://github.com/flyingday/Public/blob/master/FMCMST.pdf
[2] (2017). *APMonitor Optimization Suite*. [Online]. Available: http://apmonitor.com/
[3] S. Baruah *et al.*, "The preemptive uniprocessor scheduling of mixed-criticality implicit-deadline sporadic task systems," in *Proc. 24th Euromicro Conf. Real Time Syst.*, Pisa, Italy, 2012, pp. 145–154.
[4] S. Baruah *et al.*, "Preemptive uniprocessor scheduling of mixed-criticality sporadic task systems," *J. ACM*, vol. 62, no. 2, p. 14, 2015.
[5] S. Baruah *et al.*, "Scheduling of mixed-criticality sporadic task systems with multiple levels," in *Proc. 12th Workshop Models Algorithms Plan. Sched. Problems*, 2015, pp. 1–3.
[6] A. Burns and S. Baruah, "Towards a more practical model for mixed criticality systems," in *Proc. 1st Int. Workshop Mixed Criticality Syst.*, 2013, pp. 1–6.
[7] A. Burns and I. R. Davis, "A survey of research into mixed criticality systems," *ACM Comput. Surveys*, vol. 50, no. 6, pp. 1–37, 2017.
[8] A. Easwaran, "Demand-based scheduling of mixed-criticality sporadic tasks on one processor," in *Proc. IEEE 34th Real Time Syst. Symp. (RTSS)*, Vancouver, BC, Canada, 2013, pp. 78–87.
[9] G. Chen *et al.*, "Utilization-based scheduling of flexible mixed-criticality real-time tasks," *IEEE Trans. Comput.*, vol. 67, no. 4, pp. 543–558, Apr. 2018.
[10] X. Gu, A. Easwaran, K.-M. Phan, and I. Shin, "Resource efficient isolation mechanisms in mixed-criticality scheduling," in *Proc. 27th Euromicro Conf. Real Time Syst.*, Lund, Sweden, 2015, pp. 13–24.
[11] D. S. Hochbaum, "Complexity and algorithms for nonlinear optimization problems," *Anna. Oper. Res.*, vol. 153, no. 1, pp. 257–296, 2007.
[12] (2014). *ISO 26262:Road Vehicles*. [Online]. Available: http://www.iso.org/iso/
[13] N. Kim *et al.*, "Attacking the one-out-of-m multicore problem by combining hardware management with mixed-criticality provisioning," in *Proc. IEEE Real Time Embedded Technol. Appl. Symp. (RTAS)*, Vienna, Austria, 2016, pp. 1–12.
[14] J. Lee *et al.*, "MC-ADAPT: Adaptive task dropping in mixed-criticality scheduling," *ACM Trans. Embedded Comput. Syst.*, vol. 16, no. 5, p. 163, 2017.
[15] D. Liu *et al.*, "EDF-VD scheduling of mixed-criticality system with degraded quality guarantees," in *Proc. 32nd IEEE Real Time Syst. Symp.*, 2016, pp. 35–46.

[16] D. Maxim, R. I. Davis, L. Cucu-Grosjean, and A. Easwaran, “Probabilistic analysis for mixed criticality systems using fixed priority preemptive scheduling,” in *Proc. 25th Int. Conf. Real Time Netw. Syst. (RTNS)*, 2017, pp. 237–246.
[17] P. J. Prisaznuk, “Integrated modular avionics,” in *Proc. IEEE Nat. Aerosp. Electron. Conf.*, 1992, pp. 39–45.
[18] J. Ren and L. T. X. Phan, “Mixed-criticality scheduling on multiprocessors using task grouping,” in *Proc. 27th Euromicro Conf. Real Time Syst.*, Lund, Sweden, 2015, pp. 25–34.
[19] H. Su, N. Guan, and D. Zhu, “Service guarantee exploration for mixed-criticality systems,” in *Proc. IEEE 20th Int. Conf. Embedded Real Time Comput. Syst. Appl.*, Chongqing, China, 2014, pp. 1–10.
[20] S. Vestal, “Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance,” in *Proc. 28th IEEE Int. Real Time Syst. Symp.*, Tucson, AZ, USA, 2007, pp. 239–243.

Gang Chen received the B.E. degree in biomedical engineering, the B.S. degree in mathematics and applied mathematics, and the M.S. degree in control science and engineering from Xi’an Jiaotong University, Xi’an, China, in 2008, 2008, and 2011, respectively, and the Ph.D. degree in computer science from the Technical University of Munich, Munich, Germany, in 2016.

He is currently an Associate Professor with Northeastern University, Shenyang, China. His current research interests include mixed-criticality system, energy-aware real-time scheduling, certifiable cache architecture design, and high-performance computing.

Nan Guan received the Ph.D. degree from Uppsala University, Uppsala, Sweden, in 2013.

He is currently an Assistant Professor with Hong Kong Polytechnic University, Hong Kong. His current research interests include safe-critical cyber-physical systems, including real-time scheduling theory, worst-case execution time analysis, and formal verification techniques.

Dr. Guan was a recipient of the European Design Automation Association Outstanding Dissertation Award in 2014, the Best Paper Award of IEEE Real-Time Systems Symposium in 2009, the Best Paper Award of Design Automation and Test in Europe Conference in 2013, and the Best Poster Award in the Ph.D. forum of IEEE International Parallel and Distributed Processing Symposium in 2012 and IEEE International Conference on Embedded and Real-Time Computing Systems and Applications in 2017.

Biao Hu received the B.Sc. degree in control science and engineering from the Harbin Institute of Technology, Harbin, China, in 2010, the M.Sc. degree in control science and engineering from Tsinghua University, Beijing, China, in 2013, and the Ph.D. degree from the Department of Computer Science, Technische Universität München, Munich, Germany, in 2017. He is currently an Associate Professor with the College of Information Science and Technology, Beijing University of Chemical Technology, Beijing. His current research interests includes autonomous driving, OpenCL computing in heterogeneous system, scheduling theory in real-time systems, and safety-critical embedded systems.

Dr. Hu is a Handling Editor of the *Journal of Circuits, Systems, and Computers* (Elsevier).

Wang Yi (M’94–F’14) received the Ph.D. degree in computer science from the Chalmers University of Technology, Gothenburg, Sweden, in 1991.

He is a Chair Professor with Uppsala University, Uppsala, Sweden. He is a member of Academy of Europe (Section of Informatics). His current research interests include models, algorithms, and software tools for building and analyzing computer systems in a systematic manner to ensure predictable behaviors.

Dr. Yi was a recipient of the CAV 2013 Award for contributions to model checking of real-time systems, in particular the development of UPPAAL, the foremost tool suite for automated analysis and verification of real-time systems, the Best Paper Awards of RTSS 2015, ECRTS 2015, DATE 2013, and RTSS 2009 for his contributions to real-time systems, the Outstanding Paper Award of ECRTS 2012, and the Best Tool Paper Award of ETAPS 2002. He is on the steering committee of ESWEEK, the annual joint event for major conferences in embedded systems areas. He is also on the steering committees of ACM EMSOFT (Co-Chair), ACM LCTES, and FORMATS. He serves frequently on Technical Program Committees for a large number of conferences. He was the TPC Chair of TACAS 2001, FORMATS 2005, EMSOFT 2006, HSCC 2011, and LCTES 2012 and the Track/Topic Chair for RTSS 2008 and DATE 2012–2014.

EDF-VD Scheduling of Flexible Mixed-Criticality System With Multiple-Shot Transitions

Gang Chen, Nan Guan, Biao Hu, and Wang Yi, *Fellow, IEEE*

***Abstract*—The existing mixed-criticality (MC) real-time task models assume that once any high-criticality task overruns, *all* high-criticality jobs execute up to their most pessimistic WCET estimations simultaneously in a one-shot manner. This is very pessimistic in the sense of unnecessary resource overbooking. In this paper, we propose a more generalized mixed-critical real-time task model, called flexible MC model with multiple-shot transitions (FMC-MST), to address this problem. In FMC-MST, high-criticality tasks can transit multiple intermediate levels to handle less pessimistic overruns independently and to nonuniformly scale the deadline on each level. We develop a run-time schedulability analysis for FMC-MST under EDF-VD scheduling, in which a better tradeoff between the penalties of low-criticality tasks and the overruns of high-criticality tasks is achieved to improve the service quality of low-criticality tasks. We also develop a resource optimization technique to find resource-efficient level-insertion configurations for FMC-MST task systems under MC timing constraints. Experiments demonstrate the effectiveness of FMC-MST compared with the state-of-the-art techniques.**

***Index Terms*—EDF-VD scheduling, flexible mixed-criticality (FMC) system, multiple-shot transitions.**

Manuscript received April 3, 2018; revised June 8, 2018; accepted July 2, 2018. This work was supported in part by the National Natural Science Foundation of China under Grant 61702085, Grant 61532007, Grant 61672140, and Grant 61772123, in part by the Fundamental Research Funds for the Central Universities under Grant N161604002, in part by RGC of Hong Kong under Grant ECS-25204216 and Grant GRF-15204917, in part by the University Grants Committee of Hong Kong through Hong Kong Polytechnic University under Project 1-ZVJ2, and in part by the Ministry of Education Joint Foundation for Equipment Pre-Research under Grant 6141A020333. This article was presented in the International Conference on Embedded Software 2018 and appears as part of the ESWEEK-TCAD special issue. *(Corresponding author: Gang Chen.)*

G. Chen is with the Smart Systems Laboratory, School of Computer Science and Engineering, Northeastern University, Shenyang 110819, China, and also with the Department of Computing, Hong Kong Polytechnic University, Hong Kong (e-mail:chengang@cse.neu.edu.cn).

N. Guan is with the Department of Computing, Hong Kong Polytechnic University, Hong Kong (e-mail: nan.guan@polyu.edu.hk).

B. Hu is with the College of Information Science and Technology, Beijing University of Chemical Technology, Beijing 100029, China (e-mail: hubiao@mail.buct.edu.cn).

W. Yi is with the Department of Information Technology, Uppsala University, 75105 Uppsala, Sweden, and also with the School of Computer Science and Engineering, Northeastern University, Shenyang 110819, China (e-mail: yi@it.uu.se).

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the author.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2018.2857359

I. Introduction

Integrating applications with different criticality levels on a shared computing platform has increasingly become a common trend in the design of real-time embedded systems. Such a trend has been observed in the automotive [12] and avionics [17] industries and has led to the emergence of mixed-criticality (MC) systems. An MC task model was proposed by Vestal in his seminal paper [20] about ten years ago, wherein different WCETs are specified for each task on all existing criticality levels, with the one on a higher criticality level being more pessimistic. Since then, many techniques for analyzing and scheduling MC systems have been proposed in the real-time literature (see [7] for a comprehensive review). However, these approaches proposed in nearly a decade still share very impractical assumptions on MC task execution behavior. Specifically, once any high criticality task overruns, the following behaviors are assumed.

1) All low-criticality tasks are abandoned. It is pessimistic to immediately abandon all low-criticality tasks because low-criticality tasks require a certain timing performance as well [12], [19].
2) All high-criticality tasks are assumed to exhibit high criticality behaviors. It is overly pessimistic to bind the mode switches of all high-criticality tasks together in the analysis, as the mode switches of high-criticality tasks are naturally independent.
3) High-criticality tasks are directly transited to the most pessimistic level. This will result in unnecessary resource overbooking because high-criticality tasks rarely reach its most pessimistic WCET estimation during run-time.

A. Related Work

Some solutions have been proposed to partly resolve the above problems. In Table I, we summarize the existing solutions in relation to the three problems described above. These solutions can be broadly categorized into the following classes. The first category of research offers low-criticality tasks a certain degraded service quality when the system is in high-criticality mode. Assumptions of abandoning all low-criticality tasks are relaxed by reducing the dispatch frequency of jobs [19] or by reducing the execution budget of jobs [6], [15]. However, these studies still apply a pessimistic mode-switch strategy.

To address the first and second problems, the second category of studies offer solutions for improving performance for

TABLE I
COMPARISON OF THE EXISTING SOLUTIONS

	P_1 Graceful Degradation	P_2 Independent Mode-Switches	P_3 Multiple-Shot Transition
[19], [6], [15]	√	×	×
[10], [18], [9], [14]	√	√	×
[4], [5]	×	×	√
Our Work	√	√	√

low-criticality tasks by using group-based mode-switch strategies [10], [18]. However, these mode-switch strategies are not flexible enough because the dependencies between low-criticality and high-criticality tasks are statically determined. To relax such dependencies, a new MC model, called flexible MC (FMC) model, was recently proposed in [9], where mode-switches of high-criticality tasks are independent and the service degradation of low-criticality is dynamically updated based on the overruns of high-criticality tasks. Lee *et al.* [14] proposed an MC-ADAPT framework supporting online adaptive task dropping under task-level mode switch that involves using a similar technique. However, the third problem is not addressed in these two state-of-the-art work. In [9] and [14], high-criticality tasks always directly transit to the *most* pessimistic level, in which very pessimistic design parameters are applied.

To support multishot transitions, EDF-VD scheduling algorithm is extended to support a K-level implicit-deadline task system in [4] and [5]. However, the K-level MC task model in [4] and [5] still applies impractical assumptions. Specifically, when the system switches the mode to level k, all the tasks of criticality at least k are assumed to exhibit k-level criticality behaviors (i.e., assumption P_2). All other tasks of criticality less than k are discarded (i.e., assumption P_1).

To the best of our knowledge, no work to date has addressed the above three problems collectively. Compared to existing studies, the motivation of this paper is to find a more fine-grained transition scheme for overrun handling that captures the varying execution behaviors of high-criticality tasks. Instead of always transiting to the *most* pessimistic level, the proposed MC system can undergo intermediate levels to handle overruns with less pessimistic design parameters, such that unnecessary resource over-booking can be avoided. By doing so, a better run-time tradeoff between the penalty of low-criticality tasks and the overruns of high-criticality tasks can be achieved to improve the service quality of low-criticality tasks.

B. Contributions

In this paper, we propose an FMC model with multiple-shot transitions (FMC-MST) operating on a uni-processor platform. Rather than always switching to the *most* pessimistic level (the strategy used in [9] and [14]), the new model allows each high-criticality task to progress over multiple less pessimistic intermediate levels and to scale the deadline nonuniformly on each criticality level. Since high-criticality tasks rarely reach their pessimistic WCET estimations, FMC-MST can avoid unnecessary resource overbooking for overruns by switching high-criticality tasks to less pessimistic intermediate levels. Furthermore, FMC-MST provides a fine-grained transition scheme where mode-switches are independent with these intermediate criticality levels. The overrun of a high-criticality task only raises its own criticality level while others remain at their previous criticality levels. The minimum required low-criticality service degradation is calculated to maintain the balanced system utilization, so as to secure the additional resources requested by a level-transiting task. The contributions of this paper can be summarized as follows.

1) We propose a new EDF-VD-based scheduling for an MC model with multiple-shot transition schemes. Compared to the state-of-the-art work [9], [14], this paper provides a more generalized FMC model that allows high-criticality tasks to progress through multiple criticality levels and to scale deadlines nonuniformly.
2) We develop a run-time schedulability analysis for each independent mode-switch. To improve the service quality of low-criticality tasks, the utilization balance between low-criticality and high-criticality tasks serves as a basic principle for finding an optimal service degradation strategy for low-criticality tasks to compensate for the additional resources requested by multishot overruns of high-criticality tasks.
3) We formally prove the correctness of run-time schedulability analysis for this fine-grained transition scheme.
4) We develop a resource optimization technique that can find resource-efficient level-insertion configurations for FMC-MST task systems under MC timing constraints.

Our evaluation on randomly generated task systems shows that the performance of FMC-MST outperforms the state-of-the-art MC scheduling approaches.

II. BACKGROUND

A. FMC Implicit-Deadline Sporadic Task Model With Multiple Criticality Levels

We consider an MC sporadic task system γ as consisting of a finite collection $\{\tau_1, \tau_2, \ldots, \tau_n\}$ of n MC implicit-deadline sporadic tasks with multiple criticality levels. Each task τ_i in γ generates an infinite sequence of jobs and can be specified by a tuple $\{T_i, \chi_i, \mathcal{C}_i\}$, where:

1) T_i is the minimum job-arrival intervals;
2) χ_i is the total number of criticality levels;
3) $\mathcal{C}_i = (C_i(0), C_i(1), \ldots, C_i(\chi_i - 1))$ is a vector of the worst-case execution times (WCETs). We assume that $C_i(0) \leq C_i(1) \leq \cdots \leq C_i(\chi_i - 1)$.

For the classic dual-criticality system, high-criticality task has two criticality levels with $\chi_i = 2$ and low-criticality task has one criticality level with $\chi_i = 1$. In this paper, we consider an extended dual-criticality task system in which the concepts of high-criticality task and low-criticality task are presented as follows.

Definition 1: In an MC system with multiple criticality levels, tasks with $\chi_i >= 2$ and $\chi_i = 1$ are called high-criticality and low-criticality tasks, respectively.

According to Definition 1, we can divide task set γ into low-criticality task set γ_L and high-criticality task set γ_H. In an MC system with multiple criticality levels, high-criticality tasks are allowed to have several overrun scenarios during run-time. We denote l_i as the criticality level whereby τ_i stays during run-time, and we have $l_i = \{0, 1, 2, \ldots, \chi_i - 1\}$. The mode-switch from level $l_j - 1$ to level l_j can be defined as follows.

Definition 2 (Mode-Switch $M_j^{l_j}$ and $\hat{M}_j^{l_j}$): When high-criticality task τ_j executes for its $C_j(l_j - 1)$ time units without signaling completion, high-criticality task τ_j immediately switches from level $l_j - 1$ to level l_j. This procedure is denoted as mode-switch $M_j^{l_j}$. The closest mode-switch[1] occurring before $M_j^{l_j}$ is denoted as $\hat{M}_j^{l_j}$. For the special case of $l_j = 0$, M_j^0 denotes high-criticality task τ_j executes at level 0.

In FMC-MST, each mode-switch $M_j^{l_j}$ is independent. Mode-switch $M_j^{l_j}$ does not require other high-criticality tasks to exhibit high-criticality behavior. For low-criticality tasks, their execution budget is updated dynamically in accordance with $M_j^{l_j}$. To model the degradation of low-criticality tasks on the point of mode-switch $M_j^{l_j}$, we now introduce the concept of the service level as follows.

Definition 3 (Service Level $z_i(M_j^{l_j})$): When the system has undergone mode switch $M_j^{l_j}$, up to $z_i(M_j^{l_j}) \cdot C_i(0)$ time units can be used for the execution of τ_i in one period T_i.

In this paper, we consider implicit-deadline task systems with task period being equal to the relative deadline (i.e., $T_i = d_i$). The utilization of a task denotes the ratio of its WCET to its period. We define the utilization of task τ_i at level l_i as

$$u_i(l_i) = \frac{C_i(l_i)}{T_i} \quad l_i = \{0, 1, 2, \ldots, \chi_i - 1\}.$$

The total utilization of low-criticality task set in the initial mode (i.e., all high-criticality tasks stay at criticality level 0) is defined as $u_L(0) = \sum_{\tau_i \in \gamma_L} u_i(0)$. According to Definition 3, the degraded utilization of low-criticality tasks on mode-switch $M_j^{l_j}$ can be defined as $u_L(M_j^{l_j}) = \sum_{\tau_i \in \gamma_L} z_i(M_j^{l_j}) \cdot u_i(0)$.

In this paper, we assume that the condition of $z_i(M_j^{l_j}) \leq z_i(\hat{M}_j^{l_j})$ should hold to accommodate the resource overbooking of mode-switch $M_j^{l_j}$. Correspondingly, the system utilization reduction $\Delta u_L(M_j^{l_j})$ of low-criticality tasks on mode-switch $M_j^{l_j}$ can be computed as $u_L(M_j^{l_j}) - u_L(\hat{M}_j^{l_j})$. Since $z_i(M_j^{l_j}) \leq z_i(\hat{M}_j^{l_j})$, we have $\Delta u_L(M_j^{l_j}) \leq 0$.

Remark 1: Note that, in FMC-MST, $\Delta u_L(M_j^{l_j})$ is off-line determined to guarantee a schedulable MC system (see Section III-C). In general, we do not need to specify the settings of $z_i(M_j^{l_j})$ during off-line stage. *Any on-line strategy* on tuning $z_i(M_j^{l_j})$ can be applied as long as it can achieve the required utilization reduction $\Delta u_L(M_j^{l_j})$.

B. EDF-VD Scheduling With Nonuniform Virtual Deadlines

In this paper, we study the schedulability for FMC-MST tasks model under EDF-VD scheduling. The main idea of EDF-VD is to use reduced virtual deadlines to obtain extra slack time for jobs and further decrease the workload of high-criticality tasks after mode-switch.

In EDF-VD [3], the virtual deadlines are uniformly scaled by a single deadline scaling factor x and can be defined uniformly by $d_j^v = x \cdot d_j$. In FMC-MST, we allow *non-uniform* deadline scaling factor $x_j^{l_j}$, where $x_j^{l_j} \in (0, 1)$ is a task and criticality level dependent scaling parameter, to nonuniformly set the virtual deadline as $d_j^v(l_j) = x_j^{l_j} \cdot d_j$.

[1]In general, the closest mode-switch $\hat{M}_j^{l_j}$ before $M_j^{l_j}$ can be any task's prior mode switch.

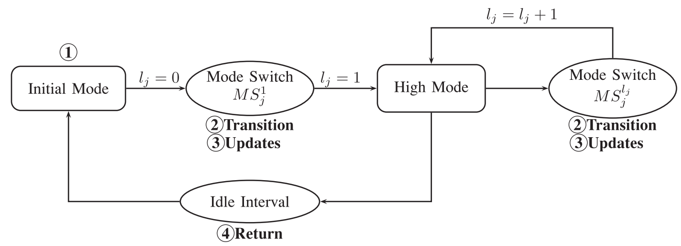

Fig. 1. Execution semantics.

C. Execution Semantics

The execution semantics of a high-criticality task is illustrated in Fig. 1. Compared to the classic MC execution model, FMC-MST model allows independent mode-switches for high-criticality tasks and dynamic service tuning for low-criticality tasks. As shown in Fig. 1, the system initially operates at level 0 (i.e., ①). An overrun of a high-criticality task only triggers itself to shift its criticality level (i.e., ②) and degrades low-criticality service to accommodate this overruns (i.e., ③). A sequence of overruns trigger the system to proceed through multiple criticality levels one by one independently (i.e., ② and ③) until the condition for transiting back is satisfied (i.e., ④). The execution semantics can be summarized as follows.

① *Initial Mode:* All tasks in γ start in level 0 (i.e., $\forall \tau_i, l_i = 0$). As long as no high-criticality task violates its $C_i(0)$, the system remains in level 0. All tasks are scheduled with $C_i(0)$.

② *Transition:* When one job of a high-criticality task τ_j *that is being executed in level* $l_j - 1$ overruns its $C_j(l_j - 1)$ without signaling completion, τ_j only triggers itself to switch into level l_j and update virtual deadline as $d_j^v(l_j)$. However, all other high-criticality tasks still stay in the same criticality level as before.

③ *Updates:* To balance the additional resource demand caused by mode-switch $M_j^{l_j}$, a new service level $z_i(M_j^{l_j})$ is determined and updated to provide degraded service for low-criticality tasks τ_i. At this updating instant, if any low-criticality jobs have completed more than $z_i(M_j^{l_j}) \cdot c_i(0)$ time units of execution, those jobs will be suspended immediately and wait for the budget to be renewed in the next period. Otherwise, low-criticality jobs can continue to use the remaining time budget for their execution.

④ *Return to Low-Criticality Mode:* When the system detects an idle interval [6], the system transits back to the low-criticality mode.

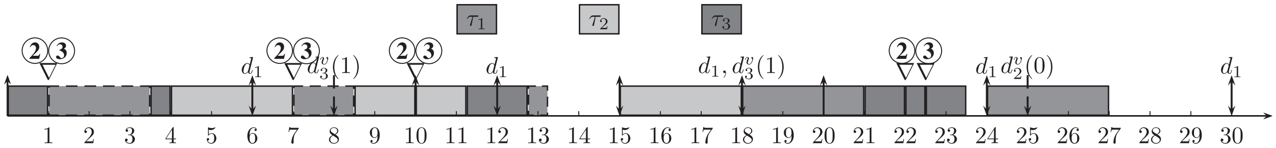

Fig. 2. Illustrative example.

TABLE II
EXAMPLE TASK SET

	χ_i	T_i	$C_i(0)/d_i^v(0)$	$C_i(1)/d_i^v(1)$	$C_i(2)/d_i^v(2)$
τ_1	1	6	3		
τ_2	3	15	$3/10$	$4.5/12.5$	$5.75/15$
τ_3	3	10	$1/5$	$1.5/8$	$2.5/10$

TABLE III
DEGRADED UTILIZATION

Mode Swithch	M_2^1	M_2^2	M_3^1	M_3^2
$\Delta u_L(M_j^{l_j})$	$-\frac{1}{6}$	$-\frac{1}{6}$	$-\frac{1}{12}$	$-\frac{1}{12}$
Budget Reduction	-1	-1	-0.5	-0.5

TABLE IV
TABLE OF NOTATIONS

Symbol	Meaning in the paper
$x_j^{l_j}$	Virtual deadline factor of task τ_j at level l_j
$u_j(l_j)$	Utilization of task τ_j at level l_j
$u_L(M_j^{l_j})$ $(u_L(\hat{M}_j^{l_j}))$	Total utilization of low-criticality tasks after (before) mode switch $M_j^{l_j}$
$\Delta u_L(M_j^{l_j})$	Utilization reduction of low-criticality tasks required to accommodate mode switch $M_j^{l_j}$
γ_H^H	Mode-switched task set $\gamma_H^H = \{\tau_j \in \gamma_H \mid l_j \geq 1\}$
γ_H^L	Non-mode-switched task set $\gamma_H^L = \{\tau_j \in \gamma_H \mid l_j = 0\}$
$a_j^{l_j}$ $(d_j^{l_j})$	Absolute release time (deadline) of the job of high-criticality τ_j that switches to $M_j^{l_j}$

D. Illustrative Example

Now, we give an example to illustrate the related concepts and execution semantics of FMC-MST. Table II gives three tasks, one low-criticality task ($\chi_1 = 1$) and two high-criticality tasks ($\chi_2 = 3$ and $\chi_3 = 3$). For high-criticality tasks, each criticality level l_j ($j = 2, 3$) associates with one virtual deadline $d_j^v(l_j)$, where $l_j \in \{0, 1, 2\}$. Table III gives the required utilization degradation $\Delta u_L(M_j^{l_j})$ for each mode switch to guarantee a schedulable MC system.[2] Fig. 2 depicts the scheduling of MC tasks under execution semantics of FMC-MST, where the symbol ∇ is used to indicate mode-switch occurrence point. In Fig. 2, the jobs are operated under the following rules.

1) Low-criticality task is scheduled with their real deadlines. In Fig. 2, τ_1 is scheduled with $d_1 = 6$.
2) At each mode switch point ∇, operation ② is triggered to update the virtual deadline while operation ③ is triggered to update the execution budget. Now we take the first mode switch as example for illustration. At $t = 1$, the first mode-switch M_3^1 occurs. τ_3 switches its criticality level from $l_3 = 0$ to $l_3 = 1$ with extending virtual deadline as $d_3^v(1) = 8$, while τ_2 stay in the same criticality level as before (i.e., ②). This deadline extension (i.e., $d_3^v(1) = 8$) simultaneously results in the pre-emption of τ_1 at $t = 1$. The execution budgets of low-criticality task τ_1 are decreased from 3 to 2.5 to achieve the required $\Delta u_L(M_3^1)$. τ_1 completes its execution at time instant 3.5 due to using up the budget (i.e., ③).
3) During a busy interval in which multiple overruns occur, the effects of the overruns on budget reduction are independent. For example, during [0, 15], three mode switches ($M_3^1 \triangleright M_2^1 \triangleright M_2^2$) occur sequentially. By Table III, the required budget reduction can be simply calculated as the sum of the one of these three mode switches, that is -2.5. Therefore, τ_1 only has 0.5 time unit for execution.

[2]The derivations for determining $\Delta u_L(M_j^{l_j})$ are illustrated in Ex 1.

III. SCHEDULABILITY ANALYSIS AND RESOURCE OPTIMIZATION

Our FMC-MST model is a more generalized model that allows multiple less pessimistic criticality levels and nonuniform deadline scaling. In this section, we present a utilization-based schedulability analysis for FMC-MST scheduling algorithm. We first analyze online schedulability for a single mode switch $M_j^{l_j}$, by which the minimum low-criticality service degradation can be derived to accommodate the resource overbooking of a mode switch. In Section III-A, we provide a high-level overview for this online schedulability analysis and attempt to communicate the intuition behind the algorithm design by means of an example. We then provide a more comprehensive description in Section III-B to prove the correctness of Theorem 1. In Section III-C, we check whether a task set is schedulable by FMC-MST under arbitrary sequences of mode switches. In Section III-D, we develop an intermediate level insertion technology and attempt to solve the problem of how to determine intermediate levels for high-criticality tasks to minimize the penalties of low-criticality tasks without sacrificing MC schedulability. We finally prove some important properties of FMC-MST. Table IV shows the notation used throughout this paper.

A. Sufficient Schedulability Test on Transition Case $M_j^{l_j}$

In this section, we provide a high-level overview of online schedulability analysis for one mode switch, and introduce the derived schedulability test condition in Theorem 1. With these conditions, we can adaptively determine how much of execution budget can be reserved for low-criticality tasks to handle each intermediate overrun while ensuring a schedulable system during run-time. Without loss of generality, we consider a general transition case $M_j^{l_j}$ where high-criticality task τ_j switches from level $l_j - 1$ to l_j, and assume the system is MC-schedulable on level $l_j - 1$. To accommodate $M_j^{l_j}$, the minimum required utilization reduction $\Delta u_L(M_j^{l_j})$ can be determined by Theorem 1.

Theorem 1: For mode-switch $M_j^{l_j}$ with $l_i \geq 1$, when high-criticality task τ_j overruns its $C_j(l_j - 1)$, the system is schedulable when the following conditions are satisfied:

$$\Delta u_L\left(M_j^{l_j}\right) + \frac{u_j(l_j)}{x_j^{l_j}} - \frac{u_j(l_j - 1)}{x_j^{l_j-1}} \leq 0 \tag{1}$$

$$\Delta u_L\left(M_j^{l_j}\right) + \frac{u_j(l_j) - u_j(l_j - 1) + p_j(l_j)}{1 - x_j^{l_j-1}} \leq 0 \tag{2}$$

$$\Delta u_L\left(M_j^{l_j}\right) \leq 0 \tag{3}$$

$$\frac{u_j(l_j)}{x_j^{l_j}} \leq u_j(\chi_j - 1) \tag{4}$$

where $p_j(l_j)$ are constrained by

$$p_j(l_j) \leq 0 \tag{5}$$

$$\sum_{l_j=1}^{\chi_j-1} p_j(l_j) = u_j(0) - \frac{u_j(0)}{x_j^0} \tag{6}$$

with the initial utilization condition on criticality level 0

$$u_L(0) + \sum_{\tau_j \in \gamma_H} \frac{u_j(0)}{x_j^0} \leq 1 \tag{7}$$

$$\frac{u_j(0)}{x_j^0} \leq u_j(\chi_j - 1). \tag{8}$$

Intuition: The intuition behind Theorem 1 is to maintain balanced system utilization during the transitions. The conditions can be explained as follows. Equation (7) ensures MC schedulability when the system stays in initial mode [3]. An event of overrun of high-criticality task normally results in an increase in virtual and overrun utilization due to resource overbooking. By analyzing the difference in virtual and overrun utilization, (1) and (2) serve as an efficient way to maintain the resource balance between the penalty of low-criticality tasks and the overruns of high-criticality tasks. Via (1) and (2), the minimum required utilization reduction $\Delta u_L(M_j^{l_j})$ can be determined to maintain the balanced system utilization, so as to secure the additional resources requested by a level-transiting task. According to [14], high-criticality task with $(u_j(l_j)/x_j^{l_j}) \geq u_j(\chi_j - 1)$ will produce schedulability loss. Therefore, additional constraints (4), (8) are imposed to avoid the performance loss during transitions. In order to provide an intuition of how the proposed analysis works, we apply Theorem 1 on a simple task set and calculate the required utilization degradation for guaranteeing MC schedulability of a single mode switch.

TABLE V
FEASIBLE SETTINGS

Mode Swithch	M_2^1	M_2^2	M_3^1	M_3^2
$p_j(l_j)$	$-\frac{2}{45}$	$-\frac{1}{18}$	$-\frac{1}{60}$	$-\frac{1}{12}$
$x_j^{l_j-1}$	$\frac{2}{3}$	$\frac{5}{6}$	$\frac{1}{2}$	$\frac{4}{5}$

Example 1: Consider a task set in Table II. Feasible settings[3] on $p_j(l_j)$ and $x_j^{l_j}$ are listed in Table V, so that conditions (4)–(8) are satisfied. In the following, we take the mode switch M_2^1 as an example to illustrate the derivation process of the required utilization degradation $\Delta u_L(M_2^1)$. According to (1)–(3) in Theorem 1, utilization degradation $\Delta u_L(M_2^1)$ should satisfy the following conditions to accommodate a feasible mode switch M_2^1:

$$\begin{aligned}\Delta u_L\left(M_2^1\right) &\\ \leq \min &\left(\underbrace{-\left(\frac{u_2(1)}{x_2^1} - \frac{u_2(0)}{x_2^0}\right)}_{\text{virtual utilization}}, \underbrace{-\frac{u_2(1) - u_2(0) + p_2(1)}{1 - x_2^0}}_{\text{overrun utilization}}, 0\right)\\ = -\frac{1}{6}.&\end{aligned}$$

The similar derivation can be operated to obtain utilization degradation for other mode switches, as presented in Table III.

B. Proof of the Correctness

We now prove the correctness of the schedulability test condition presented in Theorem 1. The proof process involves three steps. We first determine the initial conditions to ensure the schedulability of tasks in initial mode (7) and to satisfy the necessary boundary constraints (3), (4), and (8). In the second step, we prove the correctness of the sufficient condition [i.e., (1)] to ensure MC schedulability after mode switch $M_j^{l_j}$. In the third step, we propose a sufficient schedulability condition [i.e., (2)] to maintain balanced overrun utilization as the system undergoes mode transition $M_j^{l_j}$.

1) Initial Conditions: The basic assumption $z_i(M_j^{l_j}) \leq z_i(\hat{M}_j^{l_j})$ implies (3). According to [3], we can use (7) to ensure MC schedulability of in level 0. Equations (4) and (8) restrict resource utilization to levels less than those achieved in the most pessimistic level (i.e., level $\chi_j - 1$). Otherwise, tasks can directly execute in level $\chi_j - 1$ for efficient resource use [14].

2) Virtual Utilization Balance Equation: We now show how to ensure MC schedulability after mode switch $M_j^{l_j}$ occurs. This is achieved via virtual utilization balance analysis before and after mode switch $M_j^{l_j}$. By replacing the period as virtual deadline, virtual utilization of each high-criticality task τ_j on level l_j is computed as $(u_j(l_j)/x_j^{l_j})$. $u_\gamma^v(\hat{M}_j^{l_j})$ and $u_\gamma^v(M_j^{l_j})$ denote the virtual utilization of task set γ before and after mode switch $M_j^{l_j}$, respectively. To ensure the correctness of system

[3]Feasible settings can be off-line determined by formulated CSP problem presented in Section III-C.

behaviors after mode switch $M_j^{l_j}$, system virtual utilization $u_\gamma^v(M_j^{l_j})$ must meet the following condition:

$$u_\gamma^v\left(M_j^{l_j}\right) = u_L\left(M_j^{l_j}\right) + \sum_{\tau_j \in \gamma_H} \frac{u_j\left(l_j\right)}{x_j^{l_j}} \leq 1. \tag{9}$$

After mode switch $M_j^{l_j}$, high-criticality task τ_j overruns $C_j(l_j - 1)$ and shifts from level $l_j - 1$ to level l_j. With the exception of high-criticality task τ_j, all other high-criticality tasks remain at their respective criticality levels without changing the utilization. Therefore, an increase in the virtual utilization of high-criticality tasks can be determined as $(u_j(l_j)/x^{l_j}) - ([u_j(l_j-1)]/[x_j^{l_j-1}])$. For low-criticality tasks, low-criticality utilization is degraded from $u_L(\hat{M}_j^{l_j})$ to $u_L(M_j^{l_j})$ due to resource overbooking of overruns. Therefore, the difference in system virtual utilization can be formulated as

$$\begin{aligned} &u_\gamma^v\left(M_j^{l_j}\right) - u_\gamma^v\left(\hat{M}_j^{l_j}\right) \\ &\quad = \underbrace{u_L\left(M_j^{l_j}\right) - u_L\left(\hat{M}_j^{l_j}\right)}_{\text{Utilization Reduction}} + \underbrace{\frac{u_j\left(l_j\right)}{x_j^{l_j}} - \frac{u_j\left(l_j - 1\right)}{x_j^{l_j-1}}}_{\text{Utilization Increment}} \\ &\quad = \underbrace{\Delta u_L\left(M_j^{l_j}\right) + \frac{u_j\left(l_j\right)}{x_j^{l_j}} - \frac{u_j\left(l_j - 1\right)}{x_j^{l_j-1}}}_{(1)}. \end{aligned} \tag{10}$$

As the system is schedulable before mode switch $M_j^{l_j}$, we have $u_\gamma^v(\hat{M}_j^{l_j}) \leq 1$. Hence, we find that (1) ensures the correctness of $u_\gamma^v(M_j^{l_j}) \leq u_\gamma^v(\hat{M}_j^{l_j}) \leq 1$ to guarantee MC schedulability after the mode-switch.

3) Overrun Utilization Balance Equation: As the third step, we prove that the condition presented in (2) is sufficient to ensure the MC schedulability during the transition phase. We adopt the similar proof strategy based on [4] and [9] and prove it by contradiction. Suppose that there is a time interval $[0, t_f]$ such that the system undergoes mode switch $M_j^{l_j}$ and the first deadline miss occurs at t_f. Let J denote the minimal set[4] of jobs released from task set γ for which a deadline is missed. $\eta_i^{l_j}(t_1, t_2)$ denotes cumulative execution time of task τ_i when the system undergoes the mode-switch $M_j^{l_j}$ during the interval $(t_1, t_2]$. $N_\gamma^{l_j}$ denotes the sum of $\eta_i^{l_j}(0, t_f)$ for all tasks in γ. Since the first deadline miss occurs at t_f, we have $N_\gamma^{l_j} > t_f$. In the following, we will show the upper bound of $N_\gamma^{l_j}$ is less than t_f, which leads to a contradiction.

To calculate the upper bound of $N_\gamma^{l_j}$, we start the proof by introducing auxiliary lemmas to analyze the upper bound of cumulative execution time for high-criticality tasks (i.e., Lemmas 1 and 2) and low-criticality tasks (i.e., Lemma 3).

High Criticality Tasks: Since the mode switches are independent, high-criticality tasks can be divided into mode-switched task set γ_H^H and nonmode-switched task set γ_H^L. Now, we derive upper bounds of the cumulative execution time for both types of high-criticality tasks.

Lemma 1: For high-criticality task τ_j of task set γ_H^H, the cumulative execution time $\eta_j^{l_j}(0, t_f)$ can be bounded by

$$\frac{a_j^1}{x_j^0} \cdot u_j(0) + \left(t_f - a_j^1\right) u_j(1) + \sum_{r_j=2}^{l_j} \left(t_f - a_j^{r_j}\right) \Delta u_j(r_j) \tag{11}$$

where $\Delta u_j(r_j) = u_j(r_j) - u_j(r_j - 1)$.

Proof: Recall that $a_j^{r_j}$ is the absolute release time of the job executed on level r_j. High-criticality task τ_j progresses though l_j levels. Therefore, the analysis duration can be divided into $l_j + 1$ time segments, as shown in Fig. 3. During time segment $[a_j^{r_j}, a_j^{r_j+1}]$, the execution requirement per job is bounded by $c_j(r_j)$. For ease of presentation, we use $a_j^{l_j+1} = t_f$. Considering l_j time segments shown in Fig. 3, the cumulative execution time $\eta_j^{l_j}(0, t_f)$ can be bounded as

$$\begin{aligned} \eta_j^{l_j}(0, t_f) &\leq a_j^1 \cdot u_j(0) + \sum_{r_j=1}^{l_j} \left(a_j^{r_j+1} - a_j^{r_j}\right) \cdot u_j\left(r_j\right) \\ &\leq \frac{a_j^1}{x_j^0} u_j(0) + \left(a_j^2 - a_j^1\right) u_j(1) \\ &\qquad + \underline{\sum_{r_j=2}^{l_j} \left(a_j^{r_j+1} - a_j^{r_j}\right) \cdot u_j(r_j)}. \end{aligned} \tag{12}$$

Since $u_j(r_j) = \sum_{k=2}^{r_j}(u_j(k) - u_j(k-1)) + u_j(1)$, we have

$$\begin{aligned} &\sum_{r_j=2}^{l_j} \left(a_j^{r_j+1} - a_j^{r_j}\right) \cdot u_j\left(r_j\right) \\ &\quad = \left(a_j^{r_j+1} - a_j^2\right) u_j(1) + \sum_{r_j=2}^{l_j} \sum_{k=2}^{r_j} \left(a_j^{r_j+1} - a_j^{r_j}\right) \\ &\qquad \times \left(u_j(k) - u_j(k-1)\right) \\ &\quad = \left(a_j^{r_j+1} - a_j^2\right) u_j(1) + \sum_{k=2}^{l_j} \sum_{r_j=k}^{l_j} \left(a_j^{r_j+1} - a_j^{r_j}\right) \\ &\qquad \times \left(u_j(k) - u_j(k-1)\right) \\ &\quad = \left(a_j^{r_j+1} - a_j^2\right) u_j(1) + \sum_{k=2}^{l_j} \left(a_j^{l_j+1} - a_j^k\right) \left(u_j(k) - u_j(k-1)\right). \end{aligned} \tag{13}$$

Substituting the marked item in (12) with (13), $\eta_j^{l_j}(0, t_f)$ can be reformulated as

$$\begin{aligned} &\frac{a_j^1}{x_j^0} u_j(0) + \left(a_j^{l_j+1} - a_j^1\right) u_j(1) \\ &\qquad + \sum_{k=2}^{l_j} \left(a_j^{l_j+1} - a_j^k\right) \left(u_j(k) - u_j(k-1)\right). \end{aligned} \tag{14}$$

Therefore, $\eta_j^{l_j}(0, t_f)$ can be bounded as (11) by replacing $a_j^{l_j+1}$ and k with t_f and r_j, respectively. ■

[4]This minimality means that if any job is removed from J, the remainder of J will be schedulable.

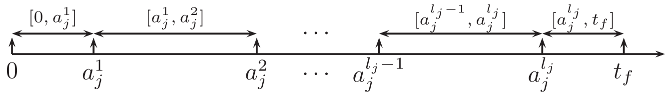

Fig. 3. Time segments.

Lemma 2 (From [9]): High-criticality task τ_j in task set γ_H^L has

$$\eta_j^0(0, t_f) \le \frac{t_f}{x_j^0} u_j(0). \tag{15}$$

Low Criticality Tasks: We now derive an upper bound on the cumulative execution time $\eta_i^{l_j}(0, t_f)$ for low-criticality tasks using a proof strategy similar to that used in [9].

Lemma 3: For low-criticality task τ_i, the cumulative execution time $\eta_i^{l_j}(0, t_f)$ can be upper bounded by

$$t_f \cdot u_i(0) + \sum_{\tau_j \in \gamma_H^H} \sum_{r_j=1}^{l_j} \psi_i^{r_j} \tag{16}$$

with *difference term* $\psi_i^{r_j} = (t_f - a_j^{r_j})(1 - x_j^{r_j-1}) \Delta u_i(M_j^{r_j})$.

Proof: We will only sketch the proof here as it is similar to the proof in [9]. The detailed proof is presented in Appendix A in the supplementary material. Following the proof strategy in [9], we analyze the difference of the cumulative execution time before and after mode-switch $M_j^{l_j}$ and prove that the difference can be uniformly upper bounded by *difference term* $\psi_i^{l_j}$. By visiting all mode switches $M_j^{r_j}$, the upper bound of $\eta_i^{l_j}(0, t_f)$ can be obtained. ■

Total Cumulative Requirements: Now, we sum the cumulative requirements over all tasks given as (17) and prove the sufficient condition (2). The complete derivation of $N_\gamma^{l_j}$ is given in Appendix B in the supplementary material

$$\begin{aligned}
N_\gamma^{l_j} &= \sum_{\tau_i \in \gamma_L} \eta_i^{l_j}(0, t_f) + \sum_{\tau_j \in \gamma_H^L} \eta_j^0(0, t_f) + \sum_{\tau_j \in \gamma_H^H} \eta_j^{l_j}(0, t_f) \\
&\le t_f + \sum_{\tau_j \in \gamma_H^H} \left(t_f - a_j^1\right) \\
&\quad \times \left(\left(1 - x_j^0\right) \Delta u_L\left(M_j^1\right) + \Delta u_j(1) + \underbrace{u_j(0) - \frac{u_j(0)}{x_j^0}}_{(6)} \right) \\
&\quad + \sum_{\tau_j \in \gamma_H^H} \sum_{r_j=2}^{l_j} \left(t_f - a_j^{r_j}\right) \left(\left(1 - x_j^{r_j-1}\right) \Delta u_L\left(M_j^{r_j}\right) + \Delta u_j(r_j) \right) \\
&= t_f + \sum_{\tau_j \in \gamma_H^H} \left(t_f - a_j^1\right) \Bigg(\left(1 - x_j^0\right) \Delta u_L\left(M_j^1\right) + \Delta u_j(1) \\
&\qquad + \sum_{l_j=1}^{\chi_j - 1} p_j(l_j) \Bigg) \\
&\quad + \sum_{\tau_j \in \gamma_H^H} \sum_{r_j=2}^{l_j} \left(t_f - a_j^{r_j}\right) \left(\left(1 - x_j^{r_j-1}\right) \Delta u_L\left(M_j^{r_j}\right) + \Delta u_j(r_j) \right)
\end{aligned}$$

Since $a_j^1 \le a_j^2 \le \cdots \le a_j^{l_j} < t_f$ and $p_j(r_j) \le 0$

$$\begin{aligned}
&\le t_f + \sum_{\tau_j \in \gamma_H^H} \sum_{r_j=1}^{l_j} \left(t_f - a_j^{r_j}\right) \Big(\left(1 - x_j^{r_j-1}\right) \Delta u_L\left(M_j^{r_j}\right) \\
&\qquad + \Delta u_j(r_j) + p_j(r_j) \Big).
\end{aligned} \tag{17}$$

The assumed deadline miss implies $N_\gamma > t_f$. That is,

$$\begin{aligned}
&\sum_{\tau_j \in \gamma_H^H} \sum_{r_j=1}^{l_j} \left(t_f - a_j^{r_j}\right) \Big(\left(1 - x_j^{r_j-1}\right) \Delta u_L\left(M_j^{r_j}\right) \\
&\qquad + \Delta u_j(r_j) + p_j(r_j) \Big) > 0.
\end{aligned}$$

Taking the contrapositive, we have

$$\begin{aligned}
&\sum_{\tau_j \in \gamma_H^H} \sum_{r_j=1}^{l_j} \left(t_f - a_j^{r_j}\right) \\
&\times \left(\underbrace{\left(1 - x_j^{r_j-1}\right) \Delta u_L\left(M_j^{r_j}\right) + \Delta u_j(r_j) + p_j(r_j)}_{(2)} \right) \le 0.
\end{aligned} \tag{18}$$

Since $t_f - a_j^{r_j} > 0$, it is sufficient to ensure the system schedulability of task set γ by guaranteeing (2) holds for each mode switch $M_j^{r_j}$. In (18), the constraints imposed on each mode switch $M_j^{r_j}$ are consistent to each other. Based on this property, the constraints imposed on current mode switch $M_j^{l_j}$ imply the condition (2), guaranteeing MC schedulability during the transition phase.

C. Feasibility of Algorithm

Theorem 1 gives an online schedulability test condition *only* for a single transition. It is yet unclear how to off-line determine whether a task set is schedulable by FMC-MST under arbitrary sequences of mode switches. In this section, we present the off-line schedulability test conditions for a task set with specified criticality levels. To guarantee schedulability, we must ensure that FMC-MST can successfully schedule the task set under any execution scenario during run-time. Therefore, to show that the task set is MC-schedulable, we need to satisfy the following two conditions.

Condition A: We need to guarantee the feasibility of each mode-switch. Therefore, constraints (1)–(8) for each mode-switch must be satisfied.

Condition B: We must ensure the system-wide feasibility. As shown in Theorem 1, each overrun will result in a decreased low-criticality utilization. For low-criticality tasks, we must show remaining low-criticality utilization should not fall below a level of 0 under the worst-case overrun scenario, that is each high-criticality task τ_j reaches criticality level $\chi_j - 1$. Therefore, we require

$$\sum_{\tau_j \in \gamma_H} \sum_{l_j=1}^{\chi_j - 1} \Delta u_L\left(M_j^{l_j}\right) + u_L(0) \ge 0. \tag{19}$$

For condition A, constraints (1)–(5) must be subjected to all mode switches with $\forall \tau_j \in \gamma_H$ and $l_j = 1, \ldots, \chi_j - 1$, while constraint (6) should be subjected to all high tasks with $\forall \tau_j \in \gamma_H$. By combining all of these conditions, we can formulate the offline schedulability problem as a constraint satisfaction problem (CSP). Any insertion solution of intermediate levels whose states satisfy a number of constraints in the derived CSP problem can guarantee a feasible scheduling system. We use the following example to illustrate how to evaluate the schedulability of the given insertion solution.

Example 2: Consider the example task set with the dedicated insertion solution given in Table II and the settings listed in Table V. We have already demonstrated condition A is satisfied, as illustrated in Example 1. For condition B, we know it is also satisfied by simply checking

$$\sum_{\tau_j \in \gamma_H} \sum_{l_j=1}^{2} \Delta u_L\left(M_j^{l_j}\right) + u_L(0) = -\frac{1}{6} - \frac{1}{6} - \frac{1}{12} - \frac{1}{12} + \frac{3}{6} = 0.$$

Therefore, the example task set in Table II is MC-schedulable.

D. Resource Optimization

Above, we prove a metric for evaluating the schedulability of an MC task set with specified level-insertion configurations. However, for an MC task set with two bounded criticality levels [i.e., $C_j(0)$ and $C_j(\chi_i - 1)$ are known], how to specify a reasonable level-insertion configuration for each high-criticality task is still not known yet. In this section, we will study the off-line resource optimization problem (ROP) with the aim of finding the resource-efficient level-insertion configuration for the FMC-MST task system within MC timing constraints.

In general, the probability that the execution time of high-criticality task reaches its most pessimistic WCET estimation is quite low. However, in EDF-VD scheduling, high-criticality tasks always transit from low-criticality level to the *most* pessimistic level once an overrun occurs. To avoid unnecessary resource over-booking, we can insert several intermediate levels to handle the less pessimistic overruns. The intermediate level to take depends on the real execution time of high-criticality tasks. In this paper, we use the distribution of the execution time of high-criticality task τ_j to compute the probability of overruns. The cumulative distribution function $F_j(t)$ is used to model the diversity of execution time of high-criticality task τ_j during run-time. Hence, the probability of the overrun $M_j^{l_j}$ that the execution time of high-criticality task τ_j falls in $[c_j(l_j), c_j(l_j + 1)]$ can be represented as $F_j(c_j(l_j + 1)) - F_j(c_j(l_j))$. When high-criticality task reaches criticality level l_j, the utilization of low-criticality tasks require to decrease $-\sum_{r_j=0}^{l_j} \Delta u_L(M_j^{r_j})$. In the off-line stage, we introduce a QoS function (20) with the aim to minimize the average low-criticality utilization decrease. Based on this objective and the aforementioned constraints, the ROP is formulated as:

$$\textbf{ROP: min} \; -\sum_{\tau_j \in \gamma_H} \sum_{l_j=1}^{\chi_j - 1} \left(F_j\left(c_j\left(l_j+1\right)\right) - F_j\left(c_j\left(l_j\right)\right)\right)$$

$$\sum_{r_j=0}^{l_j} \Delta u_L\left(M_j^{r_j}\right)$$

$$\text{s.t.} \begin{cases} \textbf{ConditionA}: & \text{Equation } (1) - (8) \\ & \text{for all mode switches} \\ \textbf{ConditionB}: & \text{Equation } (19). \end{cases} \quad (20)$$

The objective function shown above is subjected to the constraints listed in the CSP formulation (conditions A and B). Given an MC task set where two bounded execution times $[C_j(0), c_j(\chi_j - 1)]$ are specified for each high-criticality task, the resource optimization formulation can automatically generate a feasible level-insertion configuration with intermediate execution time $c_j(l_j)$ and deadline scaling factor $x_j^{l_j}$ for each high-criticality task.

Complexity: Due to nonlinear items in the constraints, the ROP (20) is a nonlinear optimization problem (NLP). For a task set with M high-criticality tasks and L criticality levels, then NLP problem has $4M(L-1) + M + 2$ constraints and $4M(L-2) + 3$ real variables. Hence, the number of variables and constraints is polynomially bounded to the size of the input problem, and it can be solved by a polynomial-time heuristic [11].

Properties: We now provide important properties to show the efficiency of FMC-MST.

Property 1: Criticality level insertions operated by ROP do not degrade the schedulability of FMC-MST.

Proof: We consider a general case that a task set is MC-schedulable by FMC-MST with L criticality levels. ROP formulation generates level-insertion configuration $[\Delta u_L(M_j^{l_j}), u_j(l_j), x_j^{l_j}, p_j(l_j)]$ for each criticality level l_j of high-criticality task τ_j. In general, without changing the previous configurations of L levels, one can insert $L+1$th level with the following configuration:

$$\Delta u_L\left(M_j^{l_j+1}\right) = 0, u_j\left(l_j+1\right) = u_j\left(l_j\right), x_j^{l_j+1} = x_j^{l_j}$$
$$p_j\left(l_j+1\right) = 0. \quad (21)$$

The new configuration still satisfies the CSP. Therefore, the task set is still MC-schedulable. ■

Property 2: FMC-MST with two criticality levels dominates EDF-AD-E [14] in terms of MC-schedulability.

Proof: For FMC-MST with two criticality levels (i.e., $\chi_j = 2$), $u_j(0)$ and $u_j(1)$ are equivalent to low-criticality and high-criticality utilization in EDF-AD-E, respectively. For task set γ, the high-criticality task set γ_H can be divided into HI-mode-preferred task set $\gamma_H^F = \{\tau_j \in \gamma_{HI} | (u_j(0)/x_j^0) \geq u_j(1)\}$ and non-HI-mode-preferred task set $\gamma_H - \gamma_H^F$, respectively. Assume task set γ is MC-schedulable by EDF-AD-E [14]. Therefore, the following conditions must be satisfied to ensure

MC schedulability according to [14]

$$u_L(0) + \min_{\tau_j \in \gamma_H} \left(\frac{u_j(0)}{x}, u_j(1) \right) \le 1 \tag{22}$$

$$x \cdot u_L(0) + u_H(1) \le 1. \tag{23}$$

In general, we can always find a lower-bound factor $\hat{x}$ that satisfies $u_L(0) + \min_{\tau_j \in \gamma_{HI}}([u_j(0)/\hat{x}], u_j(1)) = 1$ and (23).

To achieve equivalent behavior, we assign $x_j^0 = (u_j(0)/u_j(1))$ for HI-mode-preferred tasks and $\hat{x}$ for non-HI-mode-preferred tasks when applying FMC-MST. By this equivalence transformation, we can make the following observations for the CSP formulation.

1) Equations (7) and (22) are equivalent.
2) $\Delta u_L(M_j^0) = 0$ holds for HI-mode-preferred tasks.
3) For non-HI-mode-preferred tasks, the constraints (1)–(6) can be equivalently merged as (2).

Based on above observations, by (2) and (19), one can derive (23) and guarantee a feasible CSP problem for FMC-MST

$$-u_L(0) \le \sum_{\tau_j \in \gamma_H} \Delta u_L\left(M_j^1\right) \le \frac{\sum_{\tau_j \in \gamma_H - \gamma_H^F} \left(\frac{u_j(0)}{\hat{x}} - u_j(1) \right)}{1 - \hat{x}}$$

$$\rightleftharpoons -u_L(0) \le \frac{\sum_{\tau_j \in \gamma_H - \gamma_H^F} \left(\frac{u_j(0)}{\hat{x}} - u_j(1) \right)}{1 - \hat{x}}$$

$$\rightleftharpoons \hat{x} \cdot u_L(0) + u_H(1) \le u_L(0) + \sum_{\tau_j \in \gamma_H - \gamma_H^F} \frac{u_j(0)}{\hat{x}}$$

$$+ \sum_{\tau_j \in \gamma_H^F} u_j(1). \tag{24}$$

From the definition of γ_H^F and $\gamma_H - \gamma_H^F$

$$\rightleftharpoons \hat{x} \cdot u_L(0) + u_H(1) \le u_L(0) + \min_{\tau_j \in \gamma_H} \left(\frac{u_j(0)}{\hat{x}}, u_j(1) \right).$$

From the definition of $\hat{x}$

$$\rightleftharpoons \hat{x} \cdot u_L(0) + u_H(1) \le 1.$$

Therefore, we can conclude when any task set γ is MC-schedulable by EDF-AD-E [14], it is also MC-schedulable by FMC-MST with two criticality levels. ■

Property 3: FMC-MST with L criticality levels inserted by ROP dominates EDF-AD-E [14] in terms of MC-schedulability.

Proof: This can be directly proved by Properties 1 and 2. ■

IV. Evaluation

A. Experiment Setup

In this section, we conduct the simulation experiments to evaluate the effectiveness of FMC-MST by an extensive comparison to state-of-the-art approaches: EDF-AD-E [14], FMC [9], IMC [15], EDF-VD [3]. Our experiments were conducted based on randomly generated MC task systems. We adopt the same workload generation algorithm as that used in [3], [8], and [10] to randomly generate task sets with two criticality levels. In FMC-MCL, two criticality levels act as the lowest and highest criticality levels (i.e., $l_j = 0$ and $l_j = \chi_j - 1$). The resource optimization approach presented in Section III-D will automatically insert the intermediate levels between these two levels. For ease of presentation, we denote these two criticality levels as LO and HI levels during the generation process. In particular, the various parameters[5] of each task are generated in the following ways.

1) For each task τ_i, low-criticality utilization u_i^{LO} is a real number drawn at random from $[0.05, 0.15]$.[6]
2) R_i denotes the ratio of u_i^{HI}/u_i^{LO} for every high-criticality task, which is a real number drawn uniformly at random from [1, 5].
3) Task period T_i of each task is an integer drawn uniformly at random from [100, 1000].
4) pCri denotes the probability that a task τ_i is a high-criticality task, and we set it as 0.5. When τ_i is a low-criticality task, then set $C_i^{LO} = \lfloor u_i^{LO} \cdot T_i \rfloor$. Otherwise, set $C_i^{LO} = \lfloor u_i^{LO} \cdot T_i \rfloor$ and $C_i^{HI} = \lfloor u_i^{LO} \cdot R_i \cdot T_i \rfloor$.[7]

One task is generated at a time until $u_B - 0.05 \le \max\{u_{LO}^{LO} + u_{HI}^{LO}, u_{HI}^{HI}\} \le u_B$.

As stated in Remark 1, FMC-MST provides a generalized degradation strategy. For the evaluation, we adopt dropping-off strategy where low-criticality tasks are partly dropped by assigning $z_i(M_j^{l_j}) = 0$ for dropped tasks. We quantitatively compare FMC-MST with above state-of-the-art approaches in terms of offline schedulability and online performance. Following [9] and [10], online low-criticality performance is measured by the percentage of finished LC jobs (PFJ). PFJ defines the ratio of the number of finished jobs of LO-critical tasks over the total number of jobs released in a given time interval. During the simulation, the execution distribution in [16], which is a straight line on $[C_i(0), C_i(\chi_i - 1)]$ with probabilities given on a log scale, is used to generate the overrun execution time for jobs of high-criticality tasks. The system takes the intermediate level according to the actual execution time. To ensure fair comparisons, we generate a job trace for each generated task set in off-line and use this unified job trace to obtain the PFJ for all compared schemes during run-time.

B. Results

We first demonstrate the effectiveness of FMC-MST compared with state-of-the-art approaches: FMC [9], EDF-AD-E [14], IMC [15], and EDF-VD [3], in which high-criticality tasks always directly enter the most pessimistic execution mode once overrun occurs. We vary utilization bounds u_B from 0.7 to 0.95 with step size of 0.05, to evaluate offline schedulability and online performance. For FMC-MST, each high-criticality task are inserted with three intermediate levels. Each data-point was obtained by randomly generating 1000 task sets. Fig. 4 shows the acceptance ratio and average PFJ for the compared approaches. The left-axis shows PFJ values achieved for low-criticality tasks represented by the bar

[5] We also follow [13] to evaluate the performance under different settings. More results are available online [1].

[6] In FMC-MCL, u_i^{LO} and u_i^{HI} correspond to $u_i(0)$ and $u_i(\chi_j - 1)$, respectively.

[7] In FMC-MCL, C_i^{LO} and C_i^{HI} correspond to $C_i(0)$ and $C_i(\chi_j - 1)$, respectively.

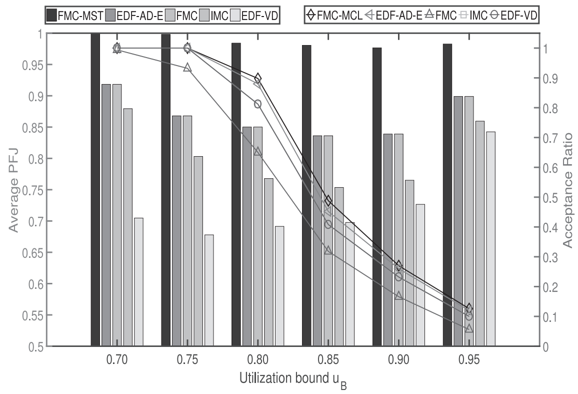

Fig. 4. Performance with varying utilization bound.

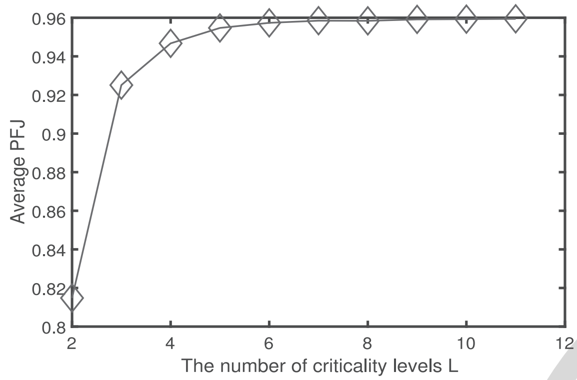

Fig. 5. Impact of the number of criticality levels L.

graphs, and the right-axis shows acceptance ratios represented by line graphs.

As shown in Fig. 4, FMC-MST can provide more low-criticality service without sacrifice in the MC schedulability. We can observe the following trends.

1) FMC-MST outperforms all the compared approaches in terms of support for low-criticality execution. This is expected because one-shot transition scheme-based approaches always switch to the level with applying the most pessimistic design parameters. In contrast, FMC-MST can capture the varying execution behavior of high-criticality tasks and can penalize low-criticality tasks more precisely according to the overrun demands of high-criticality tasks.
2) FMC-MST dominates all the EDF-VD-based scheduling algorithms with one-shot transition scheme. This schedulability performance gain is attributed to the fact that FMC-MST provides a generalized MC model where a nonuniform deadline scaling is a relaxation of EDF-VD-based schedulings [9], [14] and might cause more task sets to be deemed schedulable.

Next, we will show how the number of intermediate levels L will impact the effectiveness of FMC-MST. In this experiment, varying L from 2 to 11, we conduct the simulation on random MC task sets with $u_B = 0.85$. Fig. 5 shows online low-criticality performance under different settings on L. As shown in Fig. 5, the average PFJ increases with the number of insertion levels L. The reason for this trend is that the more insertion levels generally imply more opportunities for handling the less pessimistic overruns during run-time, which can avoid overbooking unnecessary resources.

We finally evaluate the computation time for deriving automatic intermediate level insertion by solving the formulated optimization problem presented in Section III. According to the parameters of task sets presented above, we can automatically generate an optimization problem and use the APMonitor optimization suite [2] to solve it. For all task set tested above, the selected optimization tool can generate results within 8.5 s. The results show that the formulated optimization problem can be solved efficiently.

V. Conclusion

We present a generalized FMC model that enables independent multiple-shot transitions for high-criticality tasks. A run-time schedulability test condition is successfully derived, which serves as a basis principle to find an optimal service degradation strategy for low-criticality tasks. We develop a resource optimization formulation to maximize the run-time low-criticality service quality without sacrificing MC schedulability. Experimental results illustrate the efficiency of the proposed approach.

References

[1] G. Chen *et al.* (2018). *EDF-VD Scheduling of Flexible Mixed-Criticality System With Multiple-Shot Transitions*. [Online]. Available: https://github.com/flyingday/Public/blob/master/FMCMST.pdf
[2] (2017). *APMonitor Optimization Suite*. [Online]. Available: http://apmonitor.com/
[3] S. Baruah *et al.*, "The preemptive uniprocessor scheduling of mixed-criticality implicit-deadline sporadic task systems," in *Proc. 24th Euromicro Conf. Real Time Syst.*, Pisa, Italy, 2012, pp. 145–154.
[4] S. Baruah *et al.*, "Preemptive uniprocessor scheduling of mixed-criticality sporadic task systems," *J. ACM*, vol. 62, no. 2, p. 14, 2015.
[5] S. Baruah *et al.*, "Scheduling of mixed-criticality sporadic task systems with multiple levels," in *Proc. 12th Workshop Models Algorithms Plan. Sched. Problems*, 2015, pp. 1–3.
[6] A. Burns and S. Baruah, "Towards a more practical model for mixed criticality systems," in *Proc. 1st Int. Workshop Mixed Criticality Syst.*, 2013, pp. 1–6.
[7] A. Burns and I. R. Davis, "A survey of research into mixed criticality systems," *ACM Comput. Surveys*, vol. 50, no. 6, pp. 1–37, 2017.
[8] A. Easwaran, "Demand-based scheduling of mixed-criticality sporadic tasks on one processor," in *Proc. IEEE 34th Real Time Syst. Symp. (RTSS)*, Vancouver, BC, Canada, 2013, pp. 78–87.
[9] G. Chen *et al.*, "Utilization-based scheduling of flexible mixed-criticality real-time tasks," *IEEE Trans. Comput.*, vol. 67, no. 4, pp. 543–558, Apr. 2018.
[10] X. Gu, A. Easwaran, K.-M. Phan, and I. Shin, "Resource efficient isolation mechanisms in mixed-criticality scheduling," in *Proc. 27th Euromicro Conf. Real Time Syst.*, Lund, Sweden, 2015, pp. 13–24.
[11] D. S. Hochbaum, "Complexity and algorithms for nonlinear optimization problems," *Anna. Oper. Res.*, vol. 153, no. 1, pp. 257–296, 2007.
[12] (2014). *ISO 26262:Road Vehicles*. [Online]. Available: http://www.iso.org/iso/
[13] N. Kim *et al.*, "Attacking the one-out-of-m multicore problem by combining hardware management with mixed-criticality provisioning," in *Proc. IEEE Real Time Embedded Technol. Appl. Symp. (RTAS)*, Vienna, Austria, 2016, pp. 1–12.
[14] J. Lee *et al.*, "MC-ADAPT: Adaptive task dropping in mixed-criticality scheduling," *ACM Trans. Embedded Comput. Syst.*, vol. 16, no. 5, p. 163, 2017.
[15] D. Liu *et al.*, "EDF-VD scheduling of mixed-criticality system with degraded quality guarantees," in *Proc. 32nd IEEE Real Time Syst. Symp.*, 2016, pp. 35–46.

[16] D. Maxim, R. I. Davis, L. Cucu-Grosjean, and A. Easwaran, “Probabilistic analysis for mixed criticality systems using fixed priority preemptive scheduling,” in *Proc. 25th Int. Conf. Real Time Netw. Syst. (RTNS)*, 2017, pp. 237–246.
[17] P. J. Prisaznuk, “Integrated modular avionics,” in *Proc. IEEE Nat. Aerosp. Electron. Conf.*, 1992, pp. 39–45.
[18] J. Ren and L. T. X. Phan, “Mixed-criticality scheduling on multiprocessors using task grouping,” in *Proc. 27th Euromicro Conf. Real Time Syst.*, Lund, Sweden, 2015, pp. 25–34.
[19] H. Su, N. Guan, and D. Zhu, “Service guarantee exploration for mixed-criticality systems,” in *Proc. IEEE 20th Int. Conf. Embedded Real Time Comput. Syst. Appl.*, Chongqing, China, 2014, pp. 1–10.
[20] S. Vestal, “Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance,” in *Proc. 28th IEEE Int. Real Time Syst. Symp.*, Tucson, AZ, USA, 2007, pp. 239–243.

Gang Chen received the B.E. degree in biomedical engineering, the B.S. degree in mathematics and applied mathematics, and the M.S. degree in control science and engineering from Xi’an Jiaotong University, Xi’an, China, in 2008, 2008, and 2011, respectively, and the Ph.D. degree in computer science from the Technical University of Munich, Munich, Germany, in 2016.

He is currently an Associate Professor with Northeastern University, Shenyang, China. His current research interests include mixed-criticality system, energy-aware real-time scheduling, certifiable cache architecture design, and high-performance computing.

Nan Guan received the Ph.D. degree from Uppsala University, Uppsala, Sweden, in 2013.

He is currently an Assistant Professor with Hong Kong Polytechnic University, Hong Kong. His current research interests include safe-critical cyber-physical systems, including real-time scheduling theory, worst-case execution time analysis, and formal verification techniques.

Dr. Guan was a recipient of the European Design Automation Association Outstanding Dissertation Award in 2014, the Best Paper Award of IEEE Real-Time Systems Symposium in 2009, the Best Paper Award of Design Automation and Test in Europe Conference in 2013, and the Best Poster Award in the Ph.D. forum of IEEE International Parallel and Distributed Processing Symposium in 2012 and IEEE International Conference on Embedded and Real-Time Computing Systems and Applications in 2017.

Biao Hu received the B.Sc. degree in control science and engineering from the Harbin Institute of Technology, Harbin, China, in 2010, the M.Sc. degree in control science and engineering from Tsinghua University, Beijing, China, in 2013, and the Ph.D. degree from the Department of Computer Science, Technische Universität München, Munich, Germany, in 2017. He is currently an Associate Professor with the College of Information Science and Technology, Beijing University of Chemical Technology, Beijing. His current research interests includes autonomous driving, OpenCL computing in heterogeneous system, scheduling theory in real-time systems, and safety-critical embedded systems.

Dr. Hu is a Handling Editor of the *Journal of Circuits, Systems, and Computers* (Elsevier).

Wang Yi (M’94–F’14) received the Ph.D. degree in computer science from the Chalmers University of Technology, Gothenburg, Sweden, in 1991.

He is a Chair Professor with Uppsala University, Uppsala, Sweden. He is a member of Academy of Europe (Section of Informatics). His current research interests include models, algorithms, and software tools for building and analyzing computer systems in a systematic manner to ensure predictable behaviors.

Dr. Yi was a recipient of the CAV 2013 Award for contributions to model checking of real-time systems, in particular the development of UPPAAL, the foremost tool suite for automated analysis and verification of real-time systems, the Best Paper Awards of RTSS 2015, ECRTS 2015, DATE 2013, and RTSS 2009 for his contributions to real-time systems, the Outstanding Paper Award of ECRTS 2012, and the Best Tool Paper Award of ETAPS 2002. He is on the steering committee of ESWEEK, the annual joint event for major conferences in embedded systems areas. He is also on the steering committees of ACM EMSOFT (Co-Chair), ACM LCTES, and FORMATS. He serves frequently on Technical Program Committees for a large number of conferences. He was the TPC Chair of TACAS 2001, FORMATS 2005, EMSOFT 2006, HSCC 2011, and LCTES 2012 and the Track/Topic Chair for RTSS 2008 and DATE 2012–2014.