Multicore Mixed-Criticality Systems: Partitioned Scheduling and Utilization Bound

Jian-Jun Han, *Member, IEEE*, Xin Tao, Dakai Zhu, *Member, IEEE*, Hakan Aydin, *Member, IEEE*, Zili Shao, *Member, IEEE*, and Laurence T. Yang, *Senior Member, IEEE*

***Abstract*—In mixed-criticality (MC) systems, multiple activities with various certification requirements (thus with different criticality levels) can co-exist on shared hardware platforms, where multicore processors have emerged as the *de facto* computing engines. In this paper, by using the partitioned earliest-deadline-first with virtual deadlines (EDF-VDs) scheduler for a set of periodic MC tasks running on multicore systems, we derive a *criticality-aware utilization bound* for efficient feasibility tests and then identify its *characteristics*. Our analysis shows that the bound increases with increasing number of cores and decreasing system criticality level. We show that, since the utilizations of MC tasks at different criticality levels can vary considerably, the utilization contribution of a task on different cores may have large variations and thus can significantly affect the system schedulability under the EDF-VD scheduler. Based on these observations, we propose a novel and efficient criticality-aware task partitioning algorithm (CA-TPA) to compensate for the inherent pessimism of the utilization bound. In order to improve the system schedulability, the task priorities are determined according to their utilization contributions to the system in CA-TPA. Moreover, by analyzing the utilization variations of tasks at different levels, we develop several heuristics to minimize the utilization increment and balance the workload on cores. The simulation results show that the CA-TPA scheme is very effective in achieving higher schedulability ratio and yielding balanced workloads. The actual implementation in Linux operating system further demonstrates the applicability of CA-TPA with lower run-time overhead, compared to the existing partitioning schemes.**

***Index Terms*—Embedded systems, mixed-criticality (MC), multicore systems, partitioned scheduling, utilization bound.**

Manuscript received September 25, 2016; revised January 25, 2017; accepted April 9, 2017. Date of publication April 25, 2017; date of current version December 20, 2017. This work was supported in part by the National Natural Science Foundation of China under Award 61472150, in part by the Fundamental Research Funds for the Central Universities China (HUST) under Grant 2016YXMS081 and Grant 2015TS072, and in part by the U.S. National Science Foundation under Award CNS-1422709 and Award CNS-1421855. This paper was recommended by Associate Editor J. Xue. *(Corresponding author: Xin Tao.)*

J.-J. Han, X. Tao, and L. T. Yang are with the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: jasonhan@hust.edu.cn; m201472823@hust.edu.cn; ltyang@stfx.ca).

D. Zhu is with the Department of Computer Science, University of Texas at San Antonio, San Antonio, TX 78249 USA (e-mail: dzhu@cs.utsa.edu).

H. Aydin is with the Department of Computer Science, George Mason University, Fairfax, VA 22030 USA (e-mail: aydin@cs.gmu.edu).

Z. Shao is with the Department of Computing, Hong Kong Polytechnic University, Hong Kong (e-mail: cszlshao@comp.polyu.edu.hk).

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

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.2017.2697955

I. Introduction

IN MODERN embedded systems, the ever-increasing complexity demands the integration of multiple functionalities on a common computing platform due to space, power, and cost constraints. For instance, the integrated modular avionics initiative for aerospace provides guidelines for hosting multiple avionics components on shared systems to address the increased complexity and cost [27]. In such integrated systems, diverse application activities with various certification requirements and different levels of importance (*criticality*) may co-exist. For example, the avionics certification standard DO-178 B/C defines five design assurance levels *A* to *E*, which are distinguished according to the extent of damage that result from activity failures [24]. To incorporate various certification requirements and enable the efficient management of application activities, the concept of mixed-criticality (MC) systems has been proposed in Vestal's seminal work [29]. Over the last decade, numerous MC scheduling studies have been reported for a variety of system and task models [8], [11], [12], [17], [22], [26], [30].

Unlike the traditional sporadic real-time task systems, where the worst-case resource requirements of all tasks must be satisfied, the successful execution of an MC task is defined by its own criticality level and the system's running mode. The basic principle of the MC model is to have more than one criticality level, where tasks at the kth (>1) criticality level have k different worst case execution requirements [29]. Moreover, the execution requirement of a task at $(k-1)$th level is no higher than that at kth level.

Most of the existing studies considered MC tasks running on single processor systems, with a focus on Fixed-Priority-based scheduling (FPS) [4], [9], [21], [25], [30] and earliest-deadline-first (EDF)-based scheduling techniques [7], [8], [10], [19], [28]. The most notable EDF-based scheduling algorithm for MC tasks is the recently proposed EDF with virtual deadlines (EDF-VDs) algorithm [2], [3]. The basic idea of the EDF-VD scheduler is to feasibly assign *virtual* (and *smaller*) deadlines (and thus higher priorities) to high-criticality tasks when the system operates at low-criticality mode, in order to improve the schedulability.

As multicore processors have become powerful computing engines for modern systems, there is a renewed interest in exploring scheduling algorithms for multicore/multiprocessor real-time systems. There are two classical approaches to the multiprocessor scheduling problem: 1) partitioned scheduling

and 2) global scheduling. A recent empirical study on multicore scheduling shows that, when compared to global scheduling, the partitioned-based scheduling generally has better system schedulability, where the individual job queues on processor cores and migration-less activities at run-time typically result in much lower run-time overhead [13].

Typically, the existing partitioned MC scheduling studies focus on *dual-criticality systems* (with only two criticality levels) and adopt the traditional heuristics that usually rely on either *task/system utilizations* [14], [18] [e.g., first-fit decreasing (FFD), best-fit decreasing (BFD), and worst-fit decreasing (WFD)] or the *criticality levels* [16]. It has been shown that a hybrid partitioned scheme [23], which allocates high criticality tasks using WFD and then low-criticality tasks using FFD, can effectively improve task schedulability compared to the schemes that consider only either utilization or criticality. Moreover, based on the EDF-VD algorithm and speedup factor analysis, several mapping schemes for dual-criticality systems are reported in [6]. To achieve better task schedulability (albeit at the cost of much higher time complexity), several mapping algorithms for dual-criticality systems were recently reported, such as the demand bound function (DBF)-based feasibility test [14] and the mixed integer nonlinear programming solutions that use task clustering [22].

The existing partitioned scheduling algorithms for dual-criticality systems normally consider only a task's utilization at its *highest* criticality level (i.e., its *maximum* utilization) [16], [18], [22], [23]—and this usually leads to pessimistic estimates of system utilization and thus degraded system schedulability. On the other hand, a look at the existing schedulability conditions for EDF-VD [2], [3] reveals that, in addition to its maximum utilization, an MC task's utilizations at other valid (lower) levels also play an important role.

Motivated by this observation, in this paper, we derive a utilization bound for MC tasks with *multiple criticality levels* running on multicore platforms under partitioned EDF-VD, and discuss its properties. By employing the latest variant of the EDF-VD algorithm [3], we extend our previous work [15] and propose a criticality-aware task partitioning algorithm (CA-TPA), where the utilization variations of tasks at different levels are taken into account for better schedulability and balanced workload distribution. Specifically, the contributions of this paper can be summarized as follows.

1) We develop a *criticality-aware utilization bound* for MC tasks scheduled by the partitioned EDF-VD algorithm with WFD mapping heuristic, which forms the basis of an efficient feasibility test.
2) We identify the *monotonicity* of the utilization bound, showing that the bound increases with the number of deployed processor cores and decreases with system criticality level.
3) By exploiting the variations of tasks's utilizations at *different criticality levels*, we present an efficient task partitioning algorithm with EDF-VD scheduler. Several criticality-aware heuristics are proposed to improve tasks' schedulability and balance the workload across multicores.
4) The empirical results from an actual implementation in the Linux operating system show that CA-TPA with EDF-VD is quite practical thanks to the relatively low run-time overhead and its *criticality-aware workload balancing* policy.

The remainder of this paper is organized as follows. Section II presents the task and system models and discusses the feasibility conditions of the EDF-VD scheduler. The criticality-aware utilization bound under partitioned EDF-VD with WFD is developed, and its properties are discussed, in Section III. Our CA-TPA scheme is presented in Section IV. The evaluation results and implementation are discussed in Sections V and VI concludes this paper. A review of the MC scheduling research can be found in the supplementary material.

II. System Models and Preliminaries

In this section, we first present the system and task models. The schedulability conditions for periodic MC tasks scheduled by EDF-VD on single processor are briefly reviewed, followed by the description of the problem addressed in this paper.

A. System and Task Models

We consider a multicore system that consists of $M \geq 2$ homogeneous processing cores, which are denoted as $\{\mathcal{P}_1, \ldots, \mathcal{P}_M\}$. A set of N periodic MC tasks $\Psi = \{\tau_1, \ldots, \tau_N\}$ are scheduled in the system. The MC tasks have $K > 1$ criticality levels. K is called the *system criticality level* and the system starts its operation at level-1 criticality.

An MC task τ_i is characterized by a tuple $\{\mathbf{C}_i, p_i, \ell_i\}$. ℓ_i ($1 \leq \ell_i \leq K$) indicates τ_i's criticality level (i.e., its *own* criticality). The system criticality level K is corresponds to the maximum criticality level among all the tasks. p_i denotes the task τ_i's period as well as its relative deadline (thus, we consider *implicit-deadline* task systems). The vector $\mathbf{C}_i = <c_i(1), \ldots, c_i(\ell_i)>$ represents the worst-case execution times (WCETs) of task τ_i at each criticality level, where the WCET of a task at a higher level is generally larger than that at a lower level, that is, $c_i(1) < c_i(2) < \cdots < c_i(\ell_i)$. We assume that the jth instance (job) of task τ_i arrives at time $r_{i,j} = (j-1) \cdot p_i$ and must complete its execution by its absolute deadline $d_{i,j} = j \cdot p_i$. We assume partitioned scheduling—the subset of tasks allocated to core $\mathcal{P}_m$ is denoted as Ψ_m ($m = 1, \ldots, M$) and a partition of tasks to cores is represented as $\Gamma = \{\Psi_1, \ldots, \Psi_M\}$, where $\Psi = \cup_{m=1}^{M} \Psi_m$.

We assume that the adaptive mixed criticality (AMC) scheme [3], [5] (which is applicable for both FPS and EDF-based scheduling) is adopted to manage the individual executions of jobs at run-time on the multicore system. When the current system operates at level-k ($<K$) running mode and a task τ_i executes for more than its level-k WCET $c_i(k)$ ($k < \ell_i$) without indicating its completion, the system performs a *global mode transition* and the running mode switches to level-$(k+1)$. At that moment, *all* tasks in the system with own criticality level k are discarded and no future level-k tasks are released, until the system becomes idle and gets back to level-1 running mode [6]. Once tasks are mapped to cores, we

assume that EDF-VD scheduler is deployed on each core $\mathcal{P}_m$ to schedule its subset of MC tasks Ψ_m.

B. Schedulability Conditions of EDF-VD Scheduler

The EDF-VD scheduler was first studied in the context of a single processor [2], [3] with the idea of assigning a *virtual* (and *smaller*) deadlines (thus higher priorities) to high-criticality tasks in order to improve schedulability. When the system switches to high-criticality mode, the timing requirements of high-criticality tasks can be guaranteed by restoring their original deadlines and dropping low-criticality tasks.

We first review the schedulability conditions for MC tasks running on a single processor scheduled under EDF-VD. We first introduce some key notations.

1) $u_i(k) = (c_i(k)/p_i)$*:* The utilization of task τ_i at level-k ($\leq \ell_i$).
2) $U_j(k)$*:* The *level-k utilization* of tasks at own criticality level k or higher, which is defined as
$$U_j(k) = \sum_{\forall \tau_i : \ell_i = j \wedge j \geq k} u_i(k). \tag{1}$$
3) $U_j^{\Psi_m}(k)$*:* The level-k utilization of tasks on core $\mathcal{P}_m$ at own criticality level k or higher, that is
$$U_j^{\Psi_m}(k) = \sum_{\forall \tau_i : \tau_i \in \Psi_m}^{\ell_i = j \wedge j \geq k} u_i(k). \tag{2}$$
4) $U(k)$*:* The *total* (*aggregate*) *level-k utilization* of tasks at own criticality level k or higher. Using (1), we can have
$$U(k) = \sum_{j=k}^{K} U_j(k). \tag{3}$$

By incorporating the utilizations of tasks at different criticality levels, a *sufficient* schedulability condition for implicit-deadline MC tasks scheduled under AMC-based EDF-VD algorithm was reported in [3], which is summarized in the theorem that follows.

Theorem 1 [3, Th. 3.4]: For an implicit-deadline MC task set allocated to processor core $\mathcal{P}_m$, the tasks are feasible under the EDF-VD scheduler on core $\mathcal{P}_m$ if either
$$\sum_{l=1}^{K} U_l^{\Psi_m}(l) \leq 1 \tag{4}$$
or, for some $k = 1, \ldots, K-1$, the condition below holds
$$\sum_{l=1}^{k} U_l^{\Psi_m}(l) < 1 \text{ and } \underbrace{\frac{\sum_{l=k+1}^{K} U_l^{\Psi_m}(k)}{1 - \sum_{l=1}^{k} U_l^{\Psi_m}(l)}}_{a(k)} \leq \underbrace{\frac{1 - \sum_{l=k+1}^{K} U_l^{\Psi_m}(l)}{\sum_{l=1}^{k} U_l^{\Psi_m}(l)}}_{b(k)}. \tag{5}$$

Basically, (4) states that if core $\mathcal{P}_m$ can accommodate the *maximum* utilization demands of all its tasks at their own criticality, the tasks are schedulable under EDF-VD (which actually reduces to EDF as there is no virtual deadline for any task [3]). We note that the condition is rather *pessimistic* since only the maximum utilization demands of tasks are considered.

Note that if (4) fails, $b(k) < 1$ ($k = 1, \ldots, K-1$) in (5). When (4) fails but (5) holds for some k (*condition-k*), the virtual (relative) deadline of any task τ_i on core $\mathcal{P}_m$ with own criticality level higher than k (i.e., $\ell_i > k$) can be set as $\hat{p}_i = x \cdot p_i$, where $x = [a(k), b(k)]$ (<1) is defined as a *reduction factor* for the virtual deadlines for high-criticality tasks to allow them to complete their low-criticality workloads earlier. When the system mode shifts to level-$(k+1)$ (i.e., a task exceeds its level-k WCET), *all* level-k tasks on core $\mathcal{P}_m$ are discarded, and the relative deadlines of tasks on core $\mathcal{P}_m$ with their own criticality levels higher than k will be restored to their original ones. For the detailed mechanism of virtual deadline adjustment and the discussions of EDF-VD for arbitrary-deadline dual-criticality systems (please see [3]).

Based on Theorem 1, we can obtain the following proposition related to the feasibility of MC tasks scheduled under the partitioned EDF-VD algorithm on multicore systems.

Proposition 1: For a set Ψ of MC tasks with K criticality levels running on a multicore system with M homogeneous cores, a given partition $\Gamma = \{\Psi_1, \ldots, \Psi_M\}$ is *feasible* if, Theorem 1 holds for every core $\mathcal{P}_m$ ($m = 1, \ldots, M$).

As the special case, for a dual-criticality system (i.e., $K = 2$), the task set is feasible under partitioned EDF-VD scheduler if, for each core $\mathcal{P}_m$ ($m = 1, \ldots, M$), we have
$$\text{either } U_1^{\Psi_m}(1) + U_2^{\Psi_m}(2) \leq 1 \tag{6}$$
$$\text{or } U_1^{\Psi_m}(1) < 1 \text{ and } \frac{U_2^{\Psi_m}(1)}{1 - U_1^{\Psi_m}(1)} \leq \frac{1 - U_2^{\Psi_m}(2)}{U_1^{\Psi_m}(1)}. \tag{7}$$

The MC$^K(N, M)$ Partition Problem: For a set Ψ of N MC implicit-deadline tasks with K criticality levels running on a system with M homogeneous cores, find a feasible task-to-core mapping Γ, where tasks on each core are schedulable under EDF-VD.

Clearly, when $K = 1$, the MC$^K(N, M)$ partition problem actually reduces to the classical partitioned real-time scheduling problem, which is a well-known NP-hard problem. Hence, the MC$^K(N, M)$ partition problem is NP-hard as well.

III. Criticality-Aware Utilization Bound and Its Characteristics Under Partitioned EDF-VD Scheduler

With the focus on EDF-VD for *uniprocessor systems*, Baruah *et al.* [3] identified the *speedup factor* to evaluate its optimality (i.e., how close the performance of the EDF-VD algorithm is to that of a clairvoyant algorithm): if an MC task set is schedulable by a clairvoyant algorithm, the tasks can also be feasible under EDF-VD when they are executed with a speedup factor, which is obtained by a global nonlinear continuous optimization solver (e.g., 4/3 for a dual-criticality system). To the best of our knowledge, a result on the speedup bounds for the *partitioned* EDF-VD scheduler for tasks with *multiple criticality levels* has not been reported yet.

In contrast to a speedup factor-based approach, our approach is based on deriving a utilization bound for efficient feasibility test of MC tasks with multiple levels, scheduled by partitioned EDF-VD. One prominent advantage of the utilization bound is that for a given mapping scheme, as long as the total utilization of a task set does not exceed the given bound, the partitioning generated by the scheme is guaranteed to be schedulable [20].

Note that, the schedulability conditions given in inequalities (4) and (5) are quite involved and most of the existing schemes are rather complicated (e.g., hybrid scheme [23], CA-TPA proposed in this paper, and the DBF-based heuristic [14]). Among these mapping algorithms, the simplest hybrid scheme allocates tasks to cores using different heuristics (e.g., WF and FF) according to their own criticality levels. In contrast, we adopt the simple WFD as the representative mapping heuristic to derive a utilization bound and then discuss its properties related to other task/system parameters. Despite of its simplicity, some insightful information can be obtained to guide our partitioned scheme to improve tasks' schedulability and balance the workload distribution. The detailed discussions can be found at the beginning of Section IV.

A. Level-1 Core Utilization Limit

Since task utilizations at *all* valid levels are considered, we can see that the schedulability conditions for EDF-VD expressed in inequalities (4) and (5) are quite involved. As each task has level-1 WCET, we first transform these conditions to the *simplified* schedulability condition for total level-1 utilization of tasks on any core (i.e., level-1 core utilization limit). Then, using that limit, we derive a level-1 utilization bound for partitioned EDF-VD in the next section.

Define ω as the maximum ratio of WCETs between two consecutive criticality levels for any task, i.e., $\omega = \max_{\forall \tau_i \in \Psi}\{(c_i(k+1)/c_i(k))|k = 1, \ldots, \ell_i - 1\}$, where $\omega > 1$. We can obtain the following theorem with respect to the level-1 core utilization limit for any core.

Theorem 2: For a set of periodic MC tasks Ψ_m allocated to core $\mathcal{P}_m$, the tasks are schedulable under EDF-VD if, the total level-1 utilization of the tasks $\sum_{j=1}^{K} U_j^{\Psi_m}(1)$ satisfies

$$\sum_{j=1}^{K} U_j^{\Psi_m}(1) < \lambda = \frac{K-1}{\frac{2\cdot(\omega^K-1)}{(\omega-1)^2} - \frac{2\cdot K}{\omega-1} - (K-1)^2}. \tag{8}$$

Proof: Consider the contrapositive. Suppose that *neither* inequality (4) *nor* inequality (5) holds for core $\mathcal{P}_m$. We first consider the second case, where inequality (5) fails.

For core $\mathcal{P}_m$, since inequality (4) does not hold either [i.e., $\sum_{l=1}^{K} U_l^{\Psi_m}(l) > 1$], we can have $(1 - \sum_{l=k+1}^{K} U_l^{\Psi_m}(l)/\sum_{l=1}^{k} U_l^{\Psi_m}(l)) < 1$ $(k = 1, \ldots, K-1)$ and thus the second item of inequality (5) can be transformed as $(\sum_{l=k+1}^{K} U_l^{\Psi_m}(k)/1 - \sum_{l=1}^{k} U_l^{\Psi_m}(l)) < 1$ $(k = 1, \ldots, K-1)$. Then, when inequality (5) fails for core $\mathcal{P}_m$, for each condition-k $(k = 1, \ldots, K-1)$, there is $\sum_{l=1}^{k} U_l^{\Psi_m}(l) \geq 1$ or $\sum_{l=1}^{k} U_l^{\Psi_m}(l) + \sum_{l=k+1}^{K} U_l^{\Psi_m}(k) \geq 1$. Therefore, we only need to consider the second case, that is

$$\sum_{l=1}^{k} U_l^{\Psi_m}(l) + \sum_{l=k+1}^{K} U_l^{\Psi_m}(k) \geq 1.$$

By the definition of ω, there is $U_l^{\Psi_m}(x) \leq U_l^{\Psi_m}(1) \cdot \omega^{x-1}$ $(1 \leq x \leq l \leq K)$. Hence, the above inequality is rewritten as

$$\begin{aligned}\sum_{l=1}^{k} U_l^{\Psi_m}(1) \cdot \omega^{l-1} + \sum_{l=k+1}^{K} U_l^{\Psi_m}(1) \cdot \omega^{k-1} \geq 1 \\ \sum_{j=1}^{K} U_j^{\Psi_m}(1) + \sum_{l=1}^{k} U_l^{\Psi_m}(1) \cdot (\omega^{l-1} - 1) \\ + \sum_{l=k+1}^{K} U_l^{\Psi_m}(1) \cdot \left(\omega^{k-1} - 1\right) \geq 1.\end{aligned}$$

Note that, $U_x^{\Psi_m}(1) \leq \sum_{j=1}^{K} U_j^{\Psi_m}(1)$ $(x = 1, \ldots, K)$. Then, the above inequality for condition-k is further rewritten as

$$\sum_{j=1}^{K} U_j^{\Psi_m}(1) \cdot \left(1 + \sum_{l=1}^{k}\left(\omega^{l-1} - 1\right) + \left(\omega^{k-1} - 1\right) \cdot (K-k)\right) \geq 1 \tag{9}$$

$$\Rightarrow \sum_{j=1}^{K} U_j^{\Psi_m}(1) \cdot \left(\frac{\omega^k - 1}{\omega - 1} + \omega^{k-1} \cdot (K-k) - (K-1)\right) \geq 1. \tag{10}$$

To compute $S = \sum_{k=1}^{K-1} \omega^{k-1} \cdot k$, we have

$$\begin{aligned}\omega \cdot S &= 1 \cdot \omega^1 + 2 \cdot \omega^2 + \cdots + (K-1) \cdot \omega^{K-1} \\ \omega \cdot S - S &= (K-1) \cdot \omega^{K-1} - \left(\omega^{K-2} + \omega^{K-3} + \cdots + \omega^0\right) \\ S &= \frac{(K-1) \cdot \omega^{K-1}}{\omega - 1} + \frac{1 - \omega^{K-1}}{(\omega-1)^2}.\end{aligned}$$

Adding up the above $(K-1)$ inequalities for condition-k as given in (10), we can further obtain

$$\begin{aligned}\sum_{j=1}^{K} U_j^{\Psi_m}(1) \cdot \Bigg(\frac{\omega^K - \omega}{(\omega-1)^2} - \frac{K-1}{\omega-1} + \frac{\omega^{K-1} - K}{\omega - 1} \\ - \frac{1-\omega^{K-1}}{(\omega-1)^2} - (K-1)^2\Bigg) \geq K-1 \\ \Rightarrow \sum_{j=1}^{K} U_j^{\Psi_m}(1) \geq \frac{K-1}{\frac{2\cdot(\omega^K-1)}{(\omega-1)^2} - \frac{2\cdot K}{\omega-1} - (K-1)^2} = U^{b1} = \lambda.\end{aligned}$$

We next consider the other case, where inequality (4) fails [i.e., $\sum_{l=1}^{K} U_l^{\Psi_m}(l) > 1$]. Following similar steps, we get:

$$\begin{aligned}U_1^{\Psi_m}(1) + \omega \cdot U_2^{\Psi_m}(1) + \cdots + \omega^{K-1} \cdot U_K^{\Psi_m}(1) > 1 \\ \sum_{j=1}^{K} U_j^{\Psi_m}(1) + \Big[(\omega-1) \cdot U_2^{\Psi_m}(1) + \cdots + \left(\omega^{K-1} - 1\right) \\ \times U_K^{\Psi_m}(1)\Big] > 1.\end{aligned}$$

As $U_x^{\Psi_m}(1) \le \sum_{j=1}^{K} U_j^{\Psi_m}(1)$ $(x = 1, \ldots, K)$, we have

$$\sum_{j=1}^{K} U_j^{\Psi_m}(1) + \left(\omega - 1 + \cdots + \omega^{K-1} - 1\right) \cdot \sum_{j=1}^{K} U_j^{\Psi_m}(1) > 1$$
$$\Rightarrow \sum_{j=1}^{K} U_j^{\Psi_m}(1) > \frac{1}{1 + \sum_{l=1}^{K}\left(\omega^{l-1} - 1\right)} = U^{b2}. \quad (11)$$

Define $f(k)$ $(k = 1, \ldots, K)$ in (9) as

$$f(k) = 1 + \sum_{l=1}^{k}\left(\omega^{l-1} - 1\right) + \left(\omega^{k-1} - 1\right) \cdot (K - k). \quad (12)$$

Recall that $\omega > 1$. For each k $(k = 1, \ldots, K-1)$, we can get

$$f(k+1) - f(k) = (K-k) \cdot \left(\omega^k - \omega^{k-1}\right) > 0$$
$$\Rightarrow f(k+1) > f(k).$$

Note that, by the definition of $f(k)$, there are $U^{b2} \cdot f(K) = 1$ based on (11) and $U^{b1} \cdot (f(1) + \cdots + f(K-1)) = K - 1$ based on (9). Thus, we can obtain

$$U^{b1} \cdot f(K) \cdot (K-1) > K - 1 \Rightarrow U^{b1} \cdot f(K) > 1$$
$$\Rightarrow U^{b1} > U^{b2}.$$

Therefore, if neither inequality (4) nor inequality (5) holds, we can get $\sum_{j=1}^{K} U_j^{\Psi_m}(1) \ge U^{b1} = \lambda$. Taking its contrapositive, when $\sum_{j=1}^{K} U_j^{\Psi_m}(1) < \lambda$, either (4) or (5) holds for core $\mathcal{P}_m$. Hence, the task set Ψ_m is schedulable under EDF-VD by Theorem 1. ■

Based on the level-1 core utilization limit λ under EDF-VD in (8), we can obtain the monotonic relationship between λ and other task parameters (i.e., ω and K) as follows.

Property 1: When K is fixed, the level-1 core utilization limit λ for each core scheduled by EDF-VD decreases when ω increases.

Proof: We can see that every $f(k)$ $(k = 1, \ldots, K-1)$ as defined in (12) increases when ω increases and K is fixed. As $\lambda = (K - 1/\sum_{k=1}^{K-1} f(k))$, λ decreases when ω increases. ■

Property 2: The level-1 core utilization limit λ under EDF-VD scheduler for each core decreases when the system criticality level K increases and ω is fixed.

For the proof of Property 2, please see the supplementary material.

B. Criticality-Aware Utilization Bound

Based on the level-1 core utilization limit for any core as given in (8), the minimum feasible number of tasks on any core can be found as $\beta = \lfloor(\lambda - \epsilon/\rho)\rfloor$, where $\rho = \max\{u_i(1) | i = 1, \ldots, N\}$ and ϵ is an arbitrarily small positive number. We can directly obtain the schedulability condition related to the number of MC tasks (N) running on the target system as follows.

Property 3: For a set Ψ of N periodic MC tasks with K criticality levels running on a multicore system with M cores under partitioned EDF-VD scheduler, the task set Ψ is guaranteed to be feasible as long as N does not exceed $\beta \cdot M$.

Once the number of tasks N does not exceed $\beta \cdot M$, based on the level-1 core utilization limit for any core presented in (8), the following theorem corresponding to the *level*-1 *utilization bound* can be used as an efficient feasibility test for a set of MC tasks, which execute on a multicore system scheduled under partitioned EDF-VD with the WFD mapping.

Theorem 3: For N periodic MC tasks with K criticality levels running on a multicore system with M cores, the level-1 utilization bound $U^{\text{ca,bound}}$ under partitioned EDF-VD scheduling with the WFD heuristic is

$$U^{\text{ca,bound}} = \frac{(\beta \cdot M + 1) \cdot \lambda}{\beta + 1}. \quad (13)$$

Proof: We assume that we use WFD and hence, the tasks have been sorted by their nonascending level-1 utilizations: for any two tasks τ_i and τ_j $(1 \le i < j \le N)$, $u_i(1) \ge u_j(1)$.

Suppose that the task τ_n is the first task for which the sufficient feasibility condition given in (8) fails when it is mapped to any core. Then, for each core $\mathcal{P}_m$ $(m = 1, \ldots, M)$, we get

$$\frac{c_n(1)}{p_n} + \sum_{\forall \tau_j \in \Psi_m} \frac{c_j(1)}{p_j} \ge \lambda \Rightarrow u_n(1) + \sum_{\forall \tau_j \in \Psi_m} u_j(1) \ge \lambda$$

where Ψ_m contains the subset of tasks on core $\mathcal{P}_m$ after allocating the first $(n-1)$ tasks and $|\Psi_m|$ corresponds to the number of tasks in the subset Ψ_m. Adding these M inequalities together, we have

$$(M-1) \cdot u_n(1) + \sum_{j=1}^{n} u_j(1) \ge M \cdot \lambda.$$

By the assumption that tasks have been ordered by their nonincreasing level-1 utilizations, $u_n(1) \le (\sum_{j=1}^{n} u_j(1)/n)$. Thus, the above inequality can be further rewritten as

$$\left(\frac{M-1}{n} + 1\right) \cdot \sum_{j=1}^{n} u_j(1) \ge M \cdot \lambda.$$

Hence, we can get $\sum_{j=1}^{n} u_j(1) \ge (M \cdot n \cdot \lambda / M + n - 1)$. Based on (3), considering that the total level-1 utilization of tasks $U(1) = \sum_{l=1}^{K} U_l(1) \ge \sum_{j=1}^{n} u_j(1)$, we can further have

$$U(1) \ge \frac{M \cdot n \cdot \lambda}{M + n - 1} = g(n) \quad (14)$$

where $g(n)$ is a function of n. Since neither λ nor M is related to n, the first derivative of $g(n)$ with respect to n is

$$g'(n) = \frac{M \cdot (M-1) \cdot \lambda}{(M + n - 1)^2} > 0.$$

Therefore, the minimum value of $g(n)$ can be determined when $n = \beta \cdot M + 1$, i.e., $U(1) \ge g(\beta \cdot M + 1) = ((\beta \cdot M + 1) \cdot \lambda/\beta + 1) = U^{\text{ca,bound}}$.

Hence, taking its contrapositive, when the total level-1 utilization of tasks $U(1)$ is less than $U^{\text{ca,bound}}$, WFD guarantees to generate a partition satisfying (8) for any core and thus Proposition 1 holds, which concludes the proof. ■

When the task system has no criticality certification requirement, which is denoted as non-MC task system (i.e., traditional periodic real-time task system), we can have $K = 1$, $\omega = 1$ and thus $\lambda = 1$ based on (9) and the proofs in Theorem 2. For the function $g(n)$ defined in (14) with $\lambda = 1$, we can get its first derivative as $g'(n) = (M \cdot (M-1)/(M+n-1)^2) > 0$. Therefore,

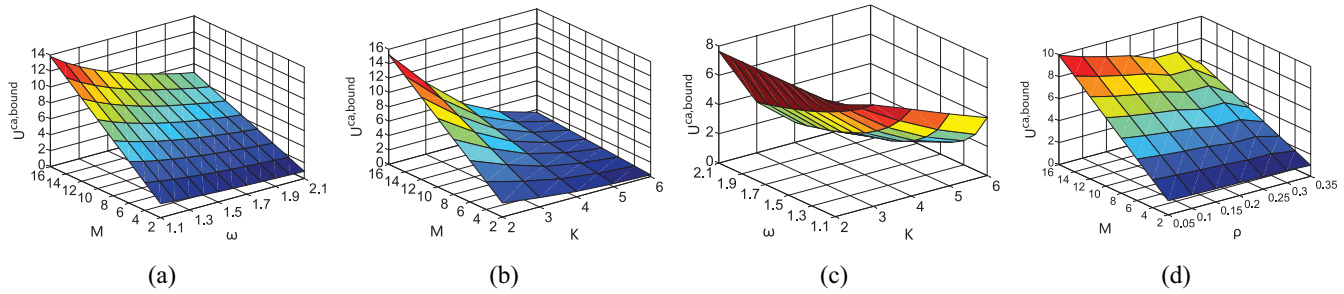


Fig. 1. Criticality-aware utilization bound under partitioned EDF-VD scheduling with the WFD heuristic. (a) $K = 3$ and $\rho = 0.1$. (b) $\omega = 1.5$ and $\rho = 0.1$. (c) $M = 8$ and $\rho = 0.1$. (d) $\omega = 1.5$ and $K = 3$.

the minimum value of $g(n)$ can be found as $(\beta \cdot M + 1/\beta + 1)$ when $n = \beta \cdot M + 1$, which actually reduces to the utilization bound for non-MC systems under partitioned-EDF with WFD [20].

Properties of the Bound: In what follows, we discuss the characteristics of the bound $U^{\text{ca,bound}}$ associated with other system/task parameters (i.e., M, ω, K, and ρ), which actually reveal the monotonicity of $U^{\text{ca,bound}}$ related to M, ω, and K. First, when the other parameters are fixed, the monotonic relationship between $U^{\text{ca,bound}}$ and the number of cores M can be identified as follows.

Property 4: For N periodic MC tasks with K criticality levels running on a multicore system, the utilization bound $U^{\text{ca,bound}}$ increases when the number of cores increases.

Proof: Let $h(M) = U^{\text{ca,bound}} = ((\beta \cdot M + 1) \cdot \lambda/\beta + 1)$. As neither β nor λ is related to M, the first derivative of $h(M)$ is $h'(M) = (\beta \cdot \lambda/\beta + 1) > 0$, which concludes the proof. ■

Next, for the parameters ω and K in λ, when other parameters are fixed, the monotonicity of the bound $U^{\text{ca,bound}}$ related to ω and K can be obtained as follows.

Property 5: For a set Ψ of MC tasks with K criticality levels running on a multicore system with M cores, the utilization bound $U^{\text{ca,bound}}$ decreases when ω and/or K increases.

Proof: Define $g(\omega, K) = \lambda$. Recall that $\beta = \lfloor(\lambda - \epsilon/\rho)\rfloor$ and let $h(\omega, K) = \beta = \lfloor(g(\omega, K) - \epsilon/\rho)\rfloor$. Then, we can have $U^{\text{ca,bound}} = ((\beta \cdot M + 1) \cdot \lambda/\beta + 1) = ((h(\omega, K) \cdot M + 1) \cdot g(\omega, K)/h(\omega, K) + 1)$.

We prove the property by contradiction. First, we assume that the claim is false. That is, there must exist two ω_1 and ω_2 ($\omega_1 > \omega_2$) such that $((h(\omega_1, K) \cdot M + 1) \cdot g(\omega_1, K)/h(\omega_1, K) + 1) \geq ((h(\omega_2, K) \cdot M + 1) \cdot g(\omega_2, K)/h(\omega_2, K) + 1)$. Based on Property 1, we have $g(\omega_1, K) < g(\omega_2, K)$ as $\omega_1 > \omega_2$. Note that ρ is not related to M, ω, K, and thus λ. Then, since the floor function of $h(\omega, K)$ (i.e., β) is considered, we can have $h(\omega_1, K) \leq h(\omega_2, K)$ and

$$\begin{aligned}
&\frac{h(\omega_1, K) \cdot M + 1}{h(\omega_1, K) + 1} \cdot g(\omega_1, K) \geq \frac{h(\omega_2, K) \cdot M + 1}{h(\omega_2, K) + 1} \cdot g(\omega_2, K) \\
&\Rightarrow \frac{h(\omega_1, K) \cdot M + 1}{h(\omega_1, K) + 1} > \frac{h(\omega_2, K) \cdot M + 1}{h(\omega_2, K) + 1} \\
&\Rightarrow h(\omega_2, K) - h(\omega_1, K) > (h(\omega_2, K) - h(\omega_1, K)) \cdot M \\
&\Rightarrow 0 > 0 \quad \text{or} \quad M < 1
\end{aligned}$$

which leads to contradiction. Following the similar steps, based on Property 2, we can also get contradictory results by assuming that $U^{\text{ca,bound}}$ does not decrease as K increases. ■

Finally, when other task/system parameters are fixed, the relationship between $U^{\text{ca,bound}}$ and ρ is given as follows.

Property 6: For a set Ψ of MC tasks with K criticality levels scheduled on a multicore system with M cores, the utilization bound $U^{\text{ca,bound}}$ cannot increase as ρ increases.

Proof: The proof is obtained by contradiction. Again, ρ is not related to M and λ. Define $h(\rho) = \beta = \lfloor(\lambda - \epsilon/\rho)\rfloor$ and we can have $U^{\text{ca,bound}} = ((h(\rho) \cdot M + 1) \cdot \lambda/h(\rho) + 1)$. Suppose that the claim is false. Then, there must exist two ρ_1 and ρ_2 ($\rho_1 > \rho_2$) such that $((h(\rho_1) \cdot M + 1) \cdot \lambda/h(\rho_1) + 1) > ((h(\rho_2) \cdot M + 1) \cdot \lambda/h(\rho_2) + 1)$. As $h(\rho)$ (i.e., β) is a floor function, $h(\rho_1) \leq h(\rho_2)$. Therefore, we can have

$$\begin{aligned}
&\frac{(h(\rho_1) \cdot M + 1) \cdot \lambda}{h(\rho_1) + 1} > \frac{(h(\rho_2) \cdot M + 1) \cdot \lambda}{h(\rho_2) + 1} \\
&\Rightarrow h(\rho_2) - h(\rho_1) > (h(\rho_2) - h(\rho_1)) \cdot M \Rightarrow 0 > 0 \text{ or } M < 1
\end{aligned}$$

which results in contradiction and concludes the proof. ■

The relationship between the criticality-aware system utilization bound $U^{\text{ca,bound}}$ and other task/system parameters (i.e., M, ω, K, and ρ) can be more explicitly illustrated in Fig. 1, where the default parameter values are: $M = 8$, $\omega = 1.5$, $K = 3$, and $\rho = 0.1$. From the figures, we can see that when there are more available cores M and smaller ω, based on Properties 4 and 5, the bound $U^{\text{ca,bound}}$ gradually increases when other parameters (i.e., K and ρ) are fixed [Fig. 1(a)]. For given M, ω, and ρ, based on Property 5, $U^{\text{ca,bound}}$ can drop dramatically when K increases [Fig. 1(b)], because higher task utilizations at high levels can lead to much smaller λ and $U^{\text{ca,bound}}$. For task systems with a high level criticality (e.g., $K \geq 4$), the bound can be extremely low, which rather limits its applicability. Similarly, based on Property 5, for given M and ρ, $U^{\text{ca,bound}}$ also decreases as ω and/or K increase as shown in Fig. 1(c). Moreover, when other parameters are fixed, $U^{\text{ca,bound}}$ exhibits stepwise decrease as ρ increases [Fig. 1(d)], which is consistent with Property 6.

IV. Criticality-Aware Task Partitioning

While $U^{\text{ca,bound}}$ provides an efficient feasibility test, it only considers the worst-case scenarios: for instance, a pessimistic ω value is used when transforming the utilizations of tasks at high criticality levels to level-1 utilizations and the WFD heuristic only takes tasks' level-1 utilizations into account.

Therefore, due to the characteristics of $U^{\text{ca,bound}}$ with respect to other task/system parameters as given in Properties 4–6, the applicability of the bound is rather limited especially when ω and/or K become large. In addition, even if the total level-1 utilization of tasks does not exceed the bound, the workload of partitions generated by WFD may be imbalanced, giving high run-time overhead as illustrated in Section V-C. More importantly, although the utilization bound (e.g., 3/4 for cumulative high-criticality utilization and low-criticality utilization on a processor [6]) can be directly employed for task mapping, such *bound-based mapping algorithm* can usually result in rather degraded schedulability even for dual-criticality systems as validated in [6].

Nonetheless, the derived bound essentially provides important insights into the partitioned scheme design for multicore MC systems with multiple criticality. The utilizations of tasks at different levels, instead of only those at a certain level [such as level-1 core utilization limit λ in the bound $U^{\text{ca,bound}}$ and the maximum utilization demands of tasks in (4)], can also affect the feasibility of tasks based on the schedulability conditions given in (4) and (5). Hence, the tasks' utilizations at *different levels* should be incorporated into the mapping heuristics for better schedulability. More importantly, as the various combinations of ω and K can generate divergent utilizations of tasks at different levels, the discrepancy among these utilizations can cause rather imbalanced workload among cores. Thus, more effective policies are needed for *criticality-aware workload balancing*, which effectively improves its applicability as validated in Section V-C.

Hence, with the objective of reducing the pessimism in the bound, the tasks' utilization variations at different levels need to be considered to enhance schedulability performance and balance system workload. Recall that the original $\text{MC}^K(N, M)$ partition problem is NP-hard. In what follows, for tasks with *multiple levels* running on multicores, we focus on efficient partitioned scheme with better practical viability (i.e., high schedulability ratio and low run-time overhead).

In general, there are two fundamental phases when mapping tasks to cores: 1) determine the order (i.e., priority) of tasks to be allocated and 2) find an appropriate core for each task. Instead of only using the maximum utilizations of tasks and the simple (but pessimistic) schedulability condition (4), we focus on the more improved schedulability condition (5). Because of the large variations of tasks' utilizations at different levels, it is crucial to take such variations into account when designing the two essential steps of partitioned scheme.

Based on the above discussions, we propose a CA-TPA. We define the *utilization contribution* of a task at a given level, which is then used to guide the allocation of tasks to cores. CA-TPA adopts a probe-based approach to ensure that, when allocating a task to cores, the overall system utilization has the *smallest increment*. Moreover, a *workload imbalance factor* is introduced to balance workload. Therefore, this paper differs from the existing studies that usually rely on only the maximum utilizations of tasks at their own criticality levels.

A. Task Ordering and Utilization Contribution

To incorporate the utilizations of tasks at different criticality levels in the first step, we present the concept of *utilization contribution* of tasks. Specifically, a task τ_i's utilization contribution at level-k ($\leq \ell_i$) is defined as

$$\mathscr{C}_i(k) = \frac{u_i(k)}{U(k)},\ k = 1, \ldots, \ell_i \tag{15}$$

where $U(k)$ defined in (3) is the total level-k utilization of tasks at the own criticality level k or higher. The utilization contribution of task τ_i to the system (by considering all its valid levels) can be further defined as

$$\mathscr{C}_i = \max\{\mathscr{C}_i(k) | k = 1, \ldots, \ell_i\}. \tag{16}$$

From the above definitions, we can see that the utilization contribution of a task essentially represents its largest weight in system utilizations in all its valid levels. Therefore, as opposed to the conventional partitioning heuristics solely based on the utilizations of tasks (such as FFD and WFD), we determine the ordering priorities of MC tasks using their utilization contributions in the first step before allocating them to cores. For this purpose, we define two relational operators $\rhd$ and $\lhd$ for task prioritization with the following rules.

1) If task τ_i has larger utilization contribution than task τ_j, we say that τ_i has higher ordering priority than τ_j (which is denoted as $\tau_i \rhd \tau_j$). Otherwise, if task τ_i's utilization contribution is smaller than that of τ_j, $\tau_i \lhd \tau_j$.
2) When the two tasks have the same utilization contribution, the tie is broken in favor of the task with higher criticality level. That is, if $\mathscr{C}_i = \mathscr{C}_j \wedge \ell_i > \ell_j$, we have $\tau_i \rhd \tau_j$.
3) If there is still a tie, the task with smaller index is assigned higher ordering priority: $\tau_i \rhd \tau_j$ if $i < j \wedge \mathscr{C}_i = \mathscr{C}_j \wedge \ell_i = \ell_j$.

B. Utilization Increment

To improve the schedulability of MC tasks while balancing workload among cores, the key point in our core selection heuristic is to take the utilization variations of tasks at different levels into consideration. Specifically, from (5), we can see that, the utilizations of tasks at *all* valid levels can affect their schedulability on a certain core. Moreover, as each core has a distinct subset of MC tasks, the utilizations of cores at each level can vary significantly [see (2)]. Therefore, the allocation of a task to different cores can lead to very divergent feasibility results and large variations in resulting core utilizations, which is quite different from the conventional partitioned scheduling of non-MC tasks.

Based on Theorem 1, once tasks are allocated to cores, we can check (4) and $(K-1)$ conditions given in (5). Note that, only one condition is required to hold on each core to satisfy the feasibility condition for the partitioned EDF-VD scheduler. In general, a partitioning scheme requires that the core utilization does not exceed 1. However, we can see that if (4) holds, $b(k) \geq 1$ for any condition-k in (5). To incorporate and exploit the core utilization to guide task mapping under

partitioned EDF-VD, condition-k in (5) can be rewritten as

$$\sum_{l=1}^{k} U_l^{\Psi_m}(l) < 1 \quad \text{and} \quad \mu^{\Psi_m}(k) \le \theta^{\Psi_m}(k)$$

$$\mu^{\Psi_m}(k) = \sum_{l=1}^{k} U_l^{\Psi_m}(l) \cdot \sum_{l=k+1}^{K} U_l^{\Psi_m}(k)$$

$$\theta^{\Psi_m}(k) = \left(1 - \sum_{l=1}^{k} U_l^{\Psi_m}(l)\right) \cdot \left(1 - \sum_{l=k+1}^{K} U_l^{\Psi_m}(l)\right) \quad (17)$$

where $\theta^{\Psi_m}(k) \le 1$ [$\theta^{\Psi_m}(k) = 1$ only if $\mathcal{P}_m$ is empty].

Intuitively, (5) [i.e., (17)] represents an improved schedulability condition compared to (4). Next, we identify the relationship between (4) and (17) regarding the schedulability. For ease of discussion, let $x(k) = \sum_{l=1}^{k} U_l^{\Psi_m}(l)$, $y(k) = \sum_{l=k+1}^{K} U_l^{\Psi_m}(l)$ and $z(k) = \sum_{l=k+1}^{K} U_l^{\Psi_m}(k)$ for condition-k on core $\mathcal{P}_m$. As $y(k)$ and $z(k)$ may be 0, we have: $y(k) \ge z(k)$ $(k = 1, \ldots, K-1)$.

Property 7: For a set Ψ of MC tasks running on a multicore system scheduled under partitioned EDF-VD, when (4) holds for core $\mathcal{P}_m$, then $\mu^{\Psi_m}(k) \le \theta^{\Psi_m}(k)$ $(k = 1, \ldots, K-1)$, that is, the second condition in (17) holds for every condition-k.

Proof: As (4) holds for core $\mathcal{P}_m$, we have $x(k) + y(k) \le 1$ $(k = 1, \ldots, K-1)$. As $y(k) \ge z(k)$, for each k

$$\begin{aligned}\theta^{\Psi_m}(k) - \mu^{\Psi_m}(k) &= [1 - x(k)] \cdot [1 - y(k)] - x(k) \cdot z(k) \\ &= 1 - [x(k) + y(k)] + x(k) \\ &\quad \times [y(k) - z(k)] \ge 0\end{aligned}$$

which concludes the proof. ∎

Property 8: For MC tasks running on multicore systems under partitioned EDF-VD, if (17) fails for any condition-k on core $\mathcal{P}_m$, (4) holds *only* when *each* task on $\mathcal{P}_m$ has its own criticality level 1 and $U_1^{\Psi_m}(1) = 1$.

Proof: As (17) fails for any condition-k $(k = 1, \ldots, K-1)$ on core $\mathcal{P}_m$, for each k, we have $x(k) \ge 1$ or $\theta^{\Psi_m}(k) - \mu^{\Psi_m}(k) < 0$. We first consider the second case

$$\begin{aligned}\theta^{\Psi_m}(k) - \mu^{\Psi_m}(k) < 0 &\Rightarrow [1 - x(k)] \cdot [1 - y(k)] \\ - x(k) \cdot z(k) < 0 &\Rightarrow \sum_{l=1}^{K} U_l^{\Psi_m}(l) = x(k) + y(k) \\ > 1 + x(k) \cdot [y(k) - z(k)] &\ge 1.\end{aligned}$$

This means that (4) also fails. Then, (17) fails but (4) holds for core $\mathcal{P}_m$ only if $x(k) \ge 1$ $(k = 1, \ldots, K-1)$. For each k, as (4) holds, we have $\sum_{l=1}^{K} U_l^{\Psi_m}(l) = x(k) + y(k) \le 1$. Then, we can get $x(k) = 1$ and $y(k) = 0$ for each k. Moreover, when $x(k) = 1$ and $y(k) = 0$ hold, $x(k+1) = 1$ and $y(k+1) = 0$ $(k = 1, \ldots, K-2)$. Hence, (17) fails but (4) holds only if $x(1) = 1$ and $y(1) = 0$, i.e., $U_1^{\Psi_m}(1) = 1$ and $U_l^{\Psi_m}(l) = 0$ $(l = 2, \ldots, K)$, which concludes the proof. ∎

Based on Properties 7 and 8, we can see that except in very rare cases, (17) represents a less pessimistic sufficient schedulability condition when compared to (4). Therefore, with an objective to incorporate task utilizations at different levels for effectively guiding the mapping of tasks to cores, (4) can be ignored and we only need to focus on the sufficient condition (17) when evaluating a core's available utilization. Then, the *available utilization for condition-k* on core $\mathcal{P}_m$ is defined as

$$A^{\Psi_m}(k) = \min\left\{\theta^{\Psi_m}(k) - \mu^{\Psi_m}(k), 1 - \sum_{l=1}^{k} U_l^{\Psi_m}(l)\right\}. \quad (18)$$

The following properties are further provided to obtain an effective available utilization for any condition-k on core $\mathcal{P}_m$.

Property 9: For a set Ψ of MC tasks with K criticality levels running on a multicore system under partitioned EDF-VD scheduler, when (4) fails but (17) holds for some k ($<K$) on core $\mathcal{P}_m$, we have $\theta^{\Psi_m}(k) - \mu^{\Psi_m}(k) < 1 - \sum_{l=1}^{k} U_l^{\Psi_m}(l)$.

Proof: Since (4) fails for core $\mathcal{P}_m$, we can have $x(k) + y(k) > 1$ $(k = 1, \ldots, K-1)$. As (17) holds for some k on core $\mathcal{P}_m$, we can get $x(k) < 1$ and

$$\begin{aligned}&\theta^{\Psi_m}(k) - \mu^{\Psi_m}(k) \ge 0 \\ \Rightarrow\ &[1 - x(k)] \cdot [1 - y(k)] - x(k) \cdot z(k) \ge 0 \\ \Rightarrow\ &1 - [x(k) + y(k)] + x(k) \cdot [y(k) - z(k)] \ge 0 \\ \Rightarrow\ &x(k) \cdot [y(k) - z(k)] > 0 \\ \Rightarrow\ &0 < x(k) < 1 \ \text{ and } \ y(k) > z(k) > 0.\end{aligned}$$

Therefore, we can have

$$\begin{aligned}&\theta^{\Psi_m}(k) - \mu^{\Psi_m}(k) - \left(1 - \sum_{l=1}^{k} U_l^{\Psi_m}(l)\right) \\ &\quad = [1 - x(k)] \cdot [1 - y(k)] - x(k) \cdot z(k) - [1 - x(k)] \\ &\quad = [x(k) - 1] \cdot y(k) - x(k) \cdot z(k) < 0\end{aligned}$$

which concludes the proof. ∎

Again, $y(k)$ and $z(k)$ may be 0 $(k = 1, \ldots, K-1)$. Following the similar steps in the proofs of Property 9, we can have the following property.

Property 10: For a set Ψ of MC tasks with K criticality levels running on a multicore system that are scheduled under partitioned EDF-VD, if (4) holds and (17) holds for some k ($<K$) on core $\mathcal{P}_m$, we have $\theta^{\Psi_m}(k) - \mu^{\Psi_m}(k) \le 1 - \sum_{l=1}^{k} U_l^{\Psi_m}(l)$.

Basically, Properties 9 and 10 indicate that no matter whether (4) holds or not, we have $\theta^{\Psi_m}(k) - \mu^{\Psi_m}(k) \le 1 - \sum_{l=1}^{k} U_l^{\Psi_m}(l)$ if condition-k holds. Hence, the available utilization for condition-k as defined in (18) can be safely determined as

$$A^{\Psi_m}(k) = \theta^{\Psi_m}(k) - \mu^{\Psi_m}(k). \quad (19)$$

Following the schedulability conditions presented in (4) and (17), the *core utilization* on core $\mathcal{P}_m$ is defined as

$$U^{\Psi_m} = \begin{cases} \infty, \sum_{l=1}^{K} U_l^{\Psi_m}(l) > 1 \ \text{ and } \ \forall k : A^{\Psi_m}(k) < 0 & \text{(20a)} \\ \quad \text{or } \sum_{l=1}^{k} U_l^{\Psi_m}(l) \ge 1, \ k = 1, \ldots, K-1 & \\ \max\{1 - A^{\Psi_m}(k) | \forall k : A^{\Psi_m}(k) \ge 0\}, \quad \text{else.} & \text{(20b)} \end{cases}$$

Essentially, (20) implies that, if the sechedulability condition given in Theorem 1 fails for core $\mathcal{P}_m$, then $U^{\Psi_m} = \infty$; otherwise, U^{Ψ_m} indicates the maximum *exploited* utilization among

all *valid* condition-k (i.e., $A^{\Psi_m}(k) \geq 0$) on core $\mathcal{P}_m$. Therefore, based on Property 7 and the definition of $A^{\Psi_m}(k)$, when (4) or (17) is satisfied for core $\mathcal{P}_m$, there must exist some k such that $A^{\Psi_m}(k) \geq 0$ holds.

Then, the *system utilization* U^{sys} and *average core utilization* U^{avg} are defined, respectively, as follows:

$$U^{\text{sys}} = \max\{U^{\Psi_m} | m = 1, \ldots, M\} \tag{21}$$

$$U^{\text{avg}} = \frac{\sum_{m=1}^{M} U^{\Psi_m}}{M}. \tag{22}$$

To further quantify the impact of partitioning a task τ_i to core $\mathcal{P}_m$, the *increment of core utilization* on $\mathcal{P}_m$ is defined as

$$\Delta^{\Psi_m \cup \{\tau_i\}} = U^{\Psi_m \cup \{\tau_i\}} - U^{\Psi_m}. \tag{23}$$

The new core utilization $U^{\Psi_m \cup \{\tau_i\}}$ of core $\mathcal{P}_m$, by assuming that task τ_i is assigned to the core, can be computed based on (20). Note that, in case $U^{\Psi_m \cup \{\tau_i\}} = \infty$ [see (20a)], τ_i cannot be feasibly allocated to core $\mathcal{P}_m$ subject to the schedulability conditions (4) and (17).

Properties in Dual-Criticality Systems: Next, focusing on dual-criticality systems (i.e., $K = 2$), we derive some properties related to the core utilization variations when allocating a task to cores. Despite the simplified assumptions, the results of these properties can shed light to design mapping heuristics and evaluate their effectiveness as discussed below.

Below, task τ_i is assumed to be allocated to core $\mathcal{P}_m$. For ease of presentation, let $x = U_1^{\Psi_m}(1)$, $y = U_2^{\Psi_m}(2)$ and for any task τ_j ($\ell_j = 2$), we have $c_j(1) = \iota \cdot c_j(2)$ where ι (<1) is a constant. Thus, $U_2^{\Psi_m}(1) = \iota \cdot U_2^{\Psi_m}(2) = \iota \cdot y$. Based on these assumptions, we can obtain the following properties related to the utilization increment after assigning τ_i to $\mathcal{P}_m$.

Property 11: Suppose that (7) holds after τ_i is mapped to $\mathcal{P}_m$. If τ_i's own criticality level equals 1 and we denote by z the quantity $u_i(1)$, then the core $\mathcal{P}_m$'s utilization increment is *no greater than* z.

Proof: For dual-criticality systems, there is only condition-1 ($k = 1$) on each core. From (7) and (20), we have

$$U^{\Psi_m} = 1 - \big[(1-x)\cdot(1-y) - \iota \cdot x \cdot y\big]$$
$$U^{\Psi_m \cup \{\tau_i\}} = 1 - \big[(1-x-z)\cdot(1-y) - \iota \cdot (x+z) \cdot y\big].$$

Note that y may be 0. Then, based on (23), we have

$$\Delta^{\Psi_m \cup \{\tau_i\}} = (1-y)\cdot z + \iota \cdot y \cdot z = \big[1 - (1-\iota)\cdot y\big] \cdot z \leq z$$

which concludes the proof. ■

Property 12: Assume that (7) holds after task τ_i is assigned to core $\mathcal{P}_m$. If task τ_i's own criticality level equals 2 and if we denote by z the quantity $u_i(2)$, the utilization increment $\Delta^{\Psi_m \cup \{\tau_i\}}$ of core $\mathcal{P}_m$ is *no greater than* z but greater than $u_i(1)$ (i.e., $\iota \cdot z$).

Proof: Following the similar steps in Property 11, as (7) holds for core $\mathcal{P}_m$, we have $0 \leq x < 1$ and

$$U^{\Psi_m \cup \{\tau_i\}} = 1 - \big[(1-x)\cdot(1-y-z) - \iota \cdot x \cdot (y+z)\big]$$
$$\Delta^{\Psi_m \cup \{\tau_i\}} = (1-x)\cdot z + \iota \cdot x \cdot z = [1 - (1-\iota)\cdot x] \cdot z \leq z.$$

Since $0 \leq x < 1$ and $0 < \iota < 1$, we can further have

$$1 - \iota > (1-\iota)\cdot x \Rightarrow 1 - (1-\iota)\cdot x > \iota \Rightarrow \Delta^{\Psi_m \cup \{\tau_i\}} > \iota \cdot z$$

which concludes the proof. ■

Based on the analysis from Properties 11 and 12, we can see that for dual-criticality systems, a lower ratio of $U_2(1)$ to $U_2(2)$ (i.e., ι) can lead to a smaller increment for core utilization, which is consistent with the intuition that more tasks with high criticality levels can reduce their deadlines to complete their relatively light low-criticality workloads earlier. More importantly, we can see that when tasks of different own criticalities are mapped onto the same core, the core utilization increment may be lower than the maximum utilization of the task to be allocated to the core.

As our partitioned scheme aims at minimizing the utilization increments of cores during task mapping, the tasks with different own criticality levels are more likely to be assigned to the same core. Therefore, when the system mode changes, the remaining high-criticality tasks can be distributed uniformly among cores, which typically results in *criticality-cognizant workload balance* and thus lower run-time overhead (due to potentially fewer job preemptions on each core) as validated in Section V-C.

Algorithm 1: Outline of CA-TPA

Input: Ψ (the task set); M (the number of cores);
Output: A feasible partition Γ or FAIL;

```
Initiate Γ = {Ψ_m}, where Ψ_m = ∅ (m = 1, ..., M);
Sort tasks in Ψ based on their utilization contributions;
for (each τ_i ∈ Ψ in the above order) do
    Δ = ∞;
    for (each P_m) do
        Calculate U^{Ψ_m ∪ {τ_i}} based on Equation (20);
        Calculate Δ^{Ψ_m ∪ {τ_i}} based on Equation (23);
        if ({Ψ_m ∪ {τ_i}} is feasible and Δ^{Ψ_m ∪ {τ_i}} < Δ) then
            Δ = Δ^{Ψ_m ∪ {τ_i}}; x = m;
        end
    end
    if (Δ == ∞) then
        Γ = ∅; break; //not feasible on any core;
    end
    Ψ_x = Ψ_x ∪ {τ_i}; //allocate τ_i to P_x;
    Update U^{Ψ_x}(k) (k = 1, ..., K) and U^{Ψ_x};
end
Return (Γ ≠ ∅ ? Γ: FAIL);
```

C. Criticality-Aware Task Partitioning Algorithm

Based on the above analysis, we adopt a probe-based approach to incorporate the contributions of tasks' utilizations at multiple levels on different cores when allocating a task to cores. Specifically, by checking all cores in the system, a task τ_i will be mapped to the core $\mathcal{P}_x$ that has the minimum increment for its core utilization, should τ_i be allocated to $\mathcal{P}_x$. That is, $\Delta^{\Psi_x \cup \{\tau_i\}} = \min\{\Delta^{\Psi_m \cup \{\tau_i\}} | m = 1, \ldots, M\}$. If more than one core has the same minimum core utilization increment, the tie is broken by mapping the task to the core with smaller index.

The outline of our CA-TPA is summarized in Algorithm 1. First, the task-to-core partition Γ and the subset of tasks for

each core are initialized (line 1). Then, all tasks are sorted in descending order of their utilization contributions (line 2). For each task, CA-TPA probes all cores by calculating its *new* core utilization [based on (20)] and utilization increment [based on (23)], by assuming that task τ_i is allocated to it (lines 5–11). For all cores that can feasibly accommodate task τ_i under the EDF-VD scheduler based on the schedulability conditions given in (4) and (17), the core $\mathcal{P}_x$ with the smallest utilization increment is chosen (line 9). If τ_i cannot be feasibly allocated to any core, CA-TPA fails to obtain a feasible task-to-core partition and quits (lines 12 and 13). Otherwise, τ_i is allocated to the target core $\mathcal{P}_x$ by updating its subset of tasks Ψ_x and related parameters (lines 15 and 16). Once all tasks have been successfully assigned to the cores, the feasible partition Γ is obtained and returned (line 18).

1) Time Complexity of CA-TPA: Recall that there are M cores and N tasks in the system. As there are normally only a few criticality levels (i.e., usually no larger than 6) in most applications, K can be assumed to be a constant in the complexity analysis. $U_j(k)$ can be computed from (1) in $O(N)$ time. The computation of $U(k)$ from (3) can also be done in $O(N)$ time. Therefore, sorting the tasks in decreasing order of their utilization contributions can be performed in the complexity of $O(N \cdot \log N)$. Next, from Algorithm 1, we can see that determining a target core that can feasibly accommodate a task can be done in $O(M+N)$ time, by computing utilization increments of all cores for all tasks that have been allocated. Hence, the overall time complexity of CA-TPA can be found as $O((M+N) \cdot N)$.

2) Workload Imbalance Factor: When tasks are allocated to cores under CA-TPA with partitioned EDF-VD, it is possible to obtain a mapping with imbalanced workloads among cores, where a few cores are over-loaded while the remaining cores have enough free capacity. To prevent our partitioning algorithm from allocating most tasks to a few cores, we introduce a *workload imbalance factor* Λ, which is defined as

$$\Lambda = \frac{U^{\text{sys}} - \min\{U^{\Psi_m} | m = 1, \ldots, M\}}{U^{\text{sys}}} \tag{24}$$

where U^{sys} is defined in (21).

In essence, Λ is exploited to control the variations of core utilizations during the task-to-core mapping. In addition, there is a threshold ρ for the workload imbalance factor Λ, which is set prior to the task assignments. When Λ increases and approaches the threshold ρ, instead of selecting a target core according to CA-TPA, the new task can be allocated directly to the valid core with the minimum core utilization (i.e., $\min\{U^{\Psi_m} | m = 1, \ldots, M\}$), subject to the feasibility conditions (4) and (17). A concrete example can be found in the supplementary material.

V. Evaluations and Discussion

To evaluate the performance of the proposed CA-TPA scheme with EDF-VD experimentally, we developed a simulator and implemented the EDF-VD scheduler in Linux kernel. For comparison, in addition to the CA-TPA scheme, we also implemented the well-known partitioning heuristics WFD, FFD, BFD, as well as the hybrid scheme proposed in [23] that can be adopted for EDF-VD and systems with *multiple criticality levels*. To test the feasibility of a core with a new task, these schemes first use the sufficient schedulability condition (4). In case the outcome is negative, they check the second and improved condition (5).

TABLE I
System and Task Parameters for the Experiments

Parameters	Values/ranges
Number of cores (M)	$[2, 32]$
System criticality level (K)	$[2, 5]$
Normalized system utilization (NSU)	$[0.4, 0.7]$
Threshold for workload imbalance (α)	$[0.1, 0.5]$
Number of tasks (N)	$[40, 200]$
Task periods (P)	$[50, 200]$,$[200, 500]$,$[500, 2000]$
Increment factor (IFC)	$[0.1, 0.7]$

As the optimal solution-based scheme reported in [22] only focuses on dual-criticality systems and is usually applicable for small problems, it is omitted here. Moreover, we also evaluate the performance of two partitioned schemes MC-P-UT-INC [6] and MPVD-HA-BF [14] pplicable to dual-criticality systems. In what follows, we first give the parameter settings for the experiments in Section V-A, and then in Sections V-B and V-C, we present and discuss the simulation results and empirical results for the tested schemes, respectively.

A. Parameter Settings

We compare these schemes based on the following performance metrics.

1) *Schedulability ratio*, which is defined as the ratio of the number of task sets that satisfy the schedulability condition to the total number of tested task sets.
2) *Average core utilization (U^{avg})* defined in (22) that assesses the workload balance of partitions generated by the schemes and can typically affect the run-time overheads for the schemes.
3) *Run-time overhead* that measures the applicability of all mapping schemes.

In Table I, we provide the parameter ranges of the system considered in the experiments, including the number of cores (M), the system criticality level (K), the normalized system utilization (NSU) (defined as the ratio of the aggregate level-1 utilization of all tasks to the number of cores), and a threshold for workload imbalance Λ (α). Then also shows the parameters for MC tasks: the number of tasks (N), task periods (P), and the increment factor (IFC) (defined as the *increasing* ratio of WCETs between two consecutive levels for any task).

In the experiments, the synthetic task sets are generated from the above parameters as follows. First, the system criticality level K is selected uniformly in the range $[2, 5]$. For given values of M, N, and NSU, the base task utilization at level-1 is set as $u_{\text{base}}(1) = (\text{NSU} \cdot M/N)$ based on the definition of NSU. Then, for each task τ_i, its period p_i is randomly chosen in one of the three period ranges given in Table I. Next, the value of $c_i(1)$ is obtained uniformly in the range $[0.2 \cdot p_i \cdot u_{\text{base}}(1), 1.8 \cdot p_i \cdot u_{\text{base}}(1)]$. The task τ_i's criticality level ℓ_i is selected uniformly within $[1, K]$. Finally, similar

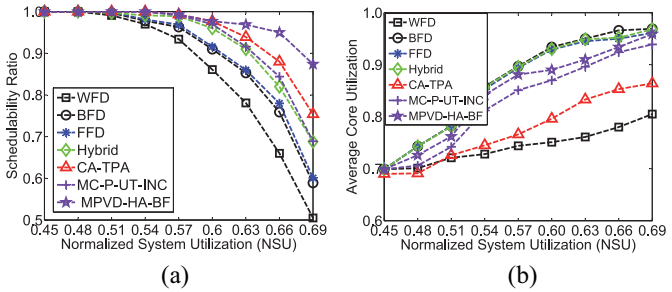


Fig. 2. Simulation performance of the schemes with varying NSU for dual-criticality systems. (a) Schedulability ratio. (b) Average core utilization.

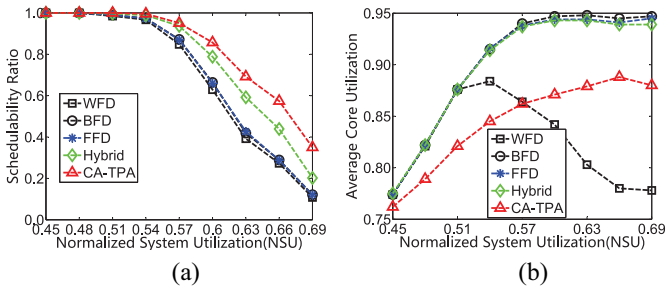


Fig. 3. Performance of the schemes with varying NSU. (a) Schedulability ratio. (b) Average core utilization.

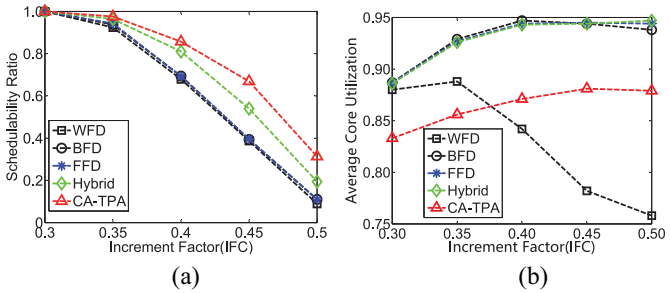


Fig. 4. Performance of the schemes with varying IFC. (a) Schedulability ratio. (b) Average core utilization.

to the generation of the value of $c_i(1)$, the values of $c_i(k)$ $(k = 2, \ldots, \ell_i)$ can be accordingly generated using $c_i(1)$ and the value of IFC.

B. Performance of the Partitioning Schemes

Unless otherwise noted, the default parameter values in the simulations are: $M = 8$, $N = 80$, $K = 4$, NSU $= 0.6$, $\alpha = 0.2$, and IFC $= 0.4$. In the reported results, each data point corresponds to the average result of 50 000 task sets.

1) Results for Dual-Criticality Systems: We first conduct the performance comparison between these mapping schemes for *dual-criticality systems* (i.e., $K = 2$). Due to space limits, we only evaluate the impact of the NSU on the performance of tested schemes and the results are shown in Fig. 2 (where IFC $= 1$).

When other parameters are fixed, larger NSU generally means higher workload and lower acceptance ratio for the schemes. Not surprisingly, as shown in Fig. 2(a), WFD usually yields the lowest acceptance ratio and CA-TPA can have the best schedulability performance among polynomial time complexity-based schemes due to its effort to minimize the core utilization increment during task-to-core mapping. MPVD-HA-BF has the best acceptance ratio but with much higher pseudo-polynomial time complexity arisen due to the use of DBF. MC-P-UT-INC considers the low-criticality workloads for high-criticality tasks and can have schedulability comparable to the CA-TPA. However, MC-P-UT-INC only focuses on dual-criticality systems and has quite high time complexity, since it iterates the values (from 0.5 to 1 in increments of 0.01 here) for the bound of the cumulative high-criticality utilization on each core to find a feasible partition.

Fig. 2(b) further shows the performance of workload balance generated by these schemes. This metric is obtained by considering only the schedulable task sets for all schemes. WFD usually generates partitions with the best workload balance among the schemes. In addition to minimizing the utilization increase of cores, CA-TPA employs a threshold for workload imbalance to avoid severely imbalanced workloads. Thus, CA-TPA can have average core utilization comparable to WFD and generate partitions with more balanced workload than other schemes.

2) Results for Systems With Multiple Criticality Levels: In what follows, we evaluate the impacts of different parameters on the performance of these partitioned schemes (except MC-P-UT-INC and MPVD-HA-BF that are applicable only to dual-criticality systems) for tasks with *multiple* criticality levels.

a) Impact of the normalized system utilization: Fig. 3 shows the impacts of the NSU on the performance for the partitioned schemes. As shown in Fig. 3(a), compared to WFD, FFD, BFD, and hybrid mapping schemes, CA-TPA can obtain much better schedulability ratio (up to 35% more) as explained above, especially when the system becomes over-loaded (e.g., NSU > 0.63). Similar to the trends as those in Fig. 2(b), Fig. 3(b) shows that CA-TPA can generate partitions with better workload balance than FFD, BFD, and hybrid, and can have lower average core utilization compared to WFD when the system is under-loaded (e.g., NSU < 0.57).

b) Impact of the increment factor: Next, we evaluate the schemes with varying IFCs and the results are shown in Fig. 4. Usually, a larger IFC causes higher system workload and lower acceptance ratio from the definition of IFC and the schedulability conditions given in Theorem 1. The results follow the similar trends as those for varying NSU: our CA-TPA-based scheme performs best in terms of schedulability ratio and generates more balanced workload than FFD, BFD, and hybrid heuristics. More specifically, as CA-TPA tries to bridge the gap between the total task utilizations at different criticality levels on every core, it can typically obtain average core utilization comparable to WFD as shown in Fig. 4(b).

c) Impact of the threshold for workload imbalance α: Fig. 5 illustrates the performance comparison among all mapping schemes with different thresholds for workload imbalance (α). As α is used only by CA-TPA to tune workload imbalance during task partitioning, the performance of other schemes remains constant when α varies as shown in Fig. 5(a) and (b).

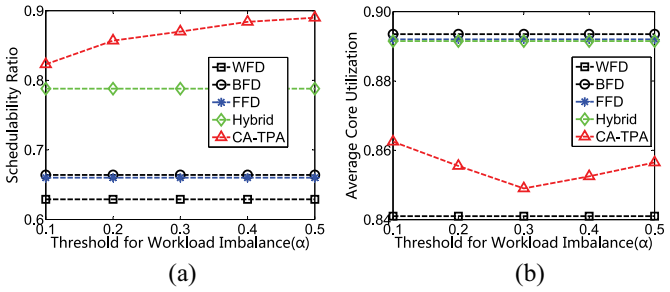


Fig. 5. Performance of the schemes with varying α. (a) Schedulability ratio. (b) Average core utilization.

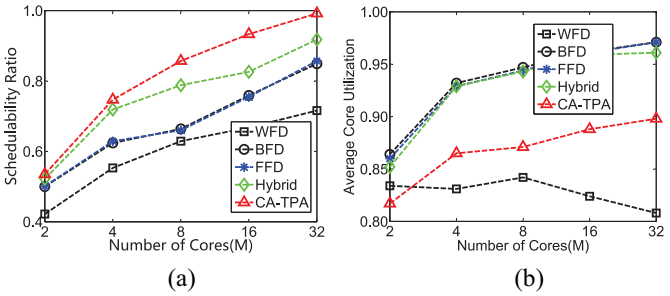


Fig. 6. Performance of the schemes with varying M. (a) Schedulability ratio. (b) Average core utilization.

A larger α usually implies larger tolerance of workload imbalance for CA-TPA. Consequently, when α increases, the CA-TPA scheme attempts to allocate tasks to the core with the minimum utilization increment without much consideration of the workload balance (i.e., in a manner similar to FFD) and thus can effectively improve schedulability as shown in Fig. 5(a). However, this behavior can result in more imbalanced workload among cores (i.e., larger workload imbalance factor), but our CA-TPA scheme still manages to generate more balanced partitions compared to FFD, BFD, and hybrid schemes as shown in Fig. 5(b).

d) Impact of the number of cores M: We further evaluate the performance for all schemes with varying number of cores (M) and the results are shown in Fig. 6. In general, more cores can provide more capacity and flexibility for tasks. Thus, when M increases, all mapping schemes can obtain better schedulability and CA-TPA still achieves the best acceptance ratio among all schemes [see Fig. 6(a)]. Due to the workload imbalance tuning during task assignments, CA-TPA can generate partitions with better workload balance compared to BFD, FFD, and hybrid [see Fig. 6(b)].

e) Impact of the number of criticality levels K: Finally, the performance comparison for all schemes with different system criticality levels (K) is shown in Fig. 7. Recall that the NSU represents the system's utilization at level-1. When other parameters are fixed, a larger K implies more execution times for tasks with highest level running at level-K. Therefore, the acceptance ratios of all schemes decrease drastically when K increases as shown in Fig. 7(a), but CA-TPA still can obtain the best schedulability performance among all schemes as explained earlier. Similarly, CA-TPA generates partitions with more balanced workload compared to BFD, FFD, and hybrid heuristics, and yields average core utilization comparable to WFD [see Fig. 7(b)].

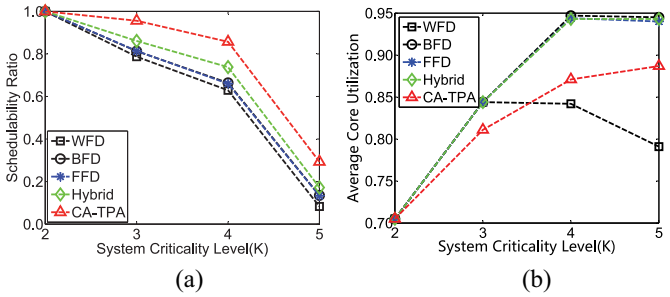


Fig. 7. Performance of the schemes with varying K. (a) Schedulability ratio. (b) Average core utilization.

C. Measurement Performance in Linux Kernel

To assess the usability of the mapping schemes, we implemented them in Linux kernel. We followed the design patterns of LITMUS^{RT} [1] and exploited the available Linux infrastructure to implement the partitioned EDF-VD scheduler: a new and highest-priority scheduling class is added to the traditional Linux scheduler and the partitioned EDF-VD scheduler always executes the highest priority jobs before the regular Linux jobs. Moreover, to maintain the highest-priority scheduling class for partitioned EDF-VD, an additional *idle* job on each core occupies CPU resource once the system is idle (i.e., the job queue on every processing core is empty).

The partitioned EDF-VD scheduler changes the Linux scheduler to invoke the initialization functions, scheduling and tick handlers at run-time. Similar to paradigms of LITMUS^{RT}, we provided a user space library to create MC tasks by means of multithreading. The tasks are initially created as non real-time where each task executes the same function codes by updating a local variable in a while-loop. A system call is utilized to pass the timing parameters of tasks from user space to kernel space, and then per-task data structures are constructed in kernel. In addition, a warm-up tick is added to ensure that the scheduling and data structures are all ready before tasks start their executions. At each hardware timer interval interrupt, every core triggers the tick handler and individually performs scheduling decisions for its jobs: including the arrival, preemption, and completion events.

Specifically, we established two *additional global synchronization mechanisms* for partitioned EDF-VD scheduler: one is used to synchronize the tick counters of all cores; based on the first mechanism and AMC scheme [5], [6], the other is used to address the issues when the system mode switches to a higher level (which is similar to barrier synchronization in [26]), such as discarding all low-criticality jobs, restoring the relative deadlines of *all* high-criticality jobs (if applicable), initializing the system running mode when it is idle.

We implemented all schemes in Linux kernel 2.6.38.8 that execute on a PC with 32 nm AMD FX-8320 processor (8 cores, 3.5 GHz clock speed, 8 MB L2, and L3 cache) and 8 GB RAM. Here, the performance metric is *total run-time*

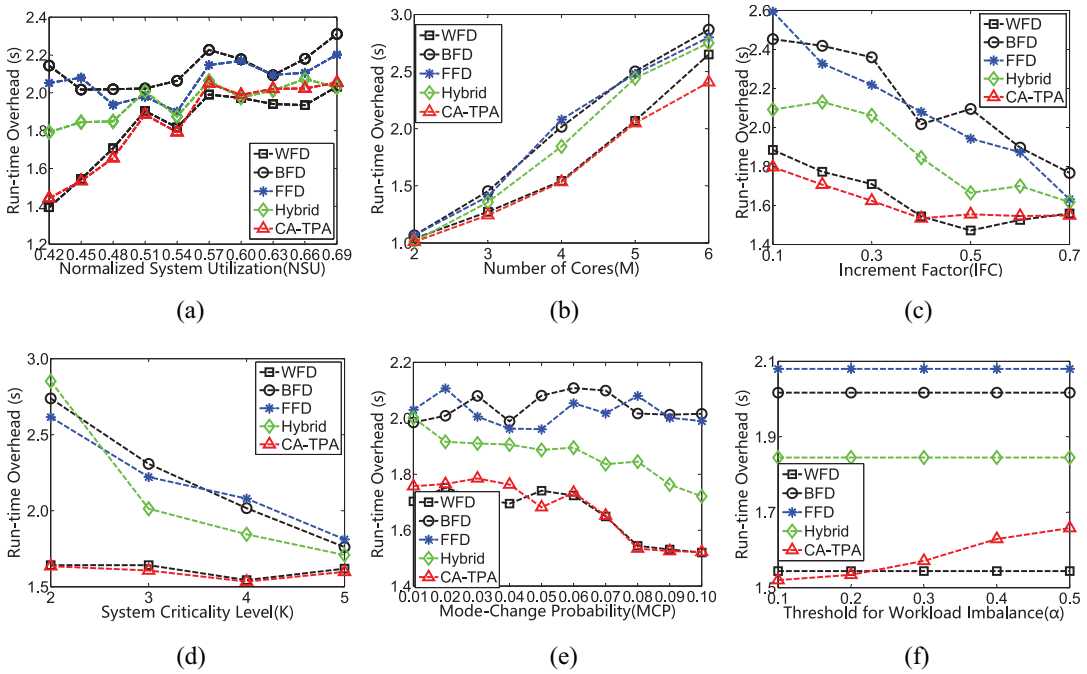


Fig. 8. Empirical performance of the task partitioning algorithms with respect to run-time overhead. (a) Performance with varying NSU. (b) Performance with varying M. (c) Performance with varying IFC. (d) Performance with varying K. (e) Performance with varying MCP. (f) Performance with varying α.

overhead on cores, where the main sources are context switching, preemption delay, operations for job queues (i.e., job arrival and job finish), and synchronization mechanisms for MC systems.

1) Parameter Settings: The period range for tasks is [50 ms, 500 ms] and other parameters are generated using the same methodology adopted in the simulations. The evaluated task sets are schedulable by *all* mapping schemes based on the feasibility conditions in (4) and (5), and each task set executes for 10 s under each scheme. The additional measuring parameter for the mapping schemes is mode-change probability (MCP), which is ranged from 0.01 to 0.1 and accounts for the execution variations of MC tasks at run-time. We first calculate the total number of jobs executed in 10 s, which is then multiplied by MCP to obtain the number of jobs that can result in mode transition. After randomly selecting such jobs (that have their own levels higher than 1), the actual execution times of these jobs can be uniformly determined from their minimum WCETs to the maximum WCETs.

Unless otherwise specified, the default parameter values in experiments are: $M = 4$, $N = 25$, $K = 4$, NSU $= 0.45$, $\alpha = 0.2$, IFC $= 0.4$, and MCP $= 0.08$. For the results reported below, each data point represents the average result of 1000 task sets.

2) Empirical Results: The overhead measurement results (in s) for the schemes are shown in Fig. 8. The overhead of CA-TPA usually corresponds about 3%–5% of the total execution time on cores (e.g., 40 s by default), which is a little lower than that (i.e., 5%) for partitioned EDF-VD on Core i5 platform [26]. Specifically, CA-TPA has measured overhead comparable to WFD and outperforms other schemes. The details of the analysis can be found in the supplementary material.

VI. Conclusion

For periodic MC tasks running on multicores under the EDF-VD algorithm, we investigated a criticality-cognizant utilization bound for partitioned EDF-VD in conjunction with the WFD heuristic, and then discussed its characteristics. We observed that as opposed to exclusively relying on tasks' maximum utilizations, the feasibility conditions for EDF-VD also depend on tasks' utilizations at other valid levels. By exploiting the contributions of tasks' utilizations at various levels on different cores, we developed a CA-TPA, and proposed several heuristics to implement task prioritization, minimize the utilization increments on cores and balance system workload. The experimental results show that compared to the existing mapping schemes, the proposed CA-TPA scheme with partitioned EDF-VD can achieve better schedulability performance with acceptable time complexity, offer more balanced partitions and experience lower run-time overhead.

References

[1] B. B. Brandenburg and J. H. Anderson, "A comparison of the M-PCP, D-PCP, and FMLP on LITMUSRT," in *Proc. Principles Distrib. Syst.*, Luxor, Egypt, 2008, pp. 105–124.

[2] 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.

[3] S. Baruah *et al.*, "Preemptive uniprocessor scheduling of mixed-criticality sporadic task systems," *J. ACM*, vol. 62, no. 2, p. 14, 2015.

[4] S. K. Baruah and A. Burns, "Fixed-priority scheduling of dual-criticality systems," in *Proc. 21st Int. Conf. Real Time Netw. Syst.*, Sophia Antipolis, France, 2013, pp. 173–181.

[5] S. K. Baruah, A. Burns, and R. I. Davis, "Response-time analysis for mixed criticality systems," in *Proc. 32nd IEEE Real Time Syst. Symp.*, Vienna, Austria, 2011, pp. 34–43.

[6] S. Baruah, B. Chattopadhyay, H. Li, and I. Shin, "Mixed-criticality scheduling on multiprocessors," *Real Time Syst.*, vol. 50, no. 1, pp. 142–177, 2014.

[7] S. Baruah and S. Vestal, "Schedulability analysis of sporadic tasks with multiple criticality specifications," in *Proc. 20th Euromicro Conf. Real Time Syst.*, 2008, pp. 147–155.

[8] Y. Chen, Q. Li, Z. Li, and H. Xiong, "Efficient schedulability analysis for mixed-criticality systems under deadline-based scheduling," *Chin. J. Aeronautics*, vol. 27, no. 4, pp. 856–866, 2014.

[9] D. De Niz, K. Lakshmanan, and R. Rajkumar, "On the scheduling of mixed-criticality real-time task sets," in *Proc. 30th IEEE Real Time Syst. Symp.*, Washington, DC, USA, 2009, pp. 291–300.

[10] P. Ekberg and W. Yi, "Bounding and shaping the demand of generalized mixed-criticality sporadic task systems," *Real Time Syst.*, vol. 50, no. 1, pp. 48–86, 2014.

[11] P. Ekberg and W. Yi, "Schedulability analysis of a graph-based task model for mixed-criticality systems," *Real Time Syst.*, vol. 52, no. 1, pp. 1–37, 2016.

[12] G. Giannopoulou, N. Stoimenov, P. Huang, L. Thiele, and B. D. de Dinechin, "Mixed-criticality scheduling on cluster-based manycores with shared communication and storage resources," *Real Time Syst.*, vol. 52, no. 4, pp. 399–449, 2016.

[13] G. Gracioli, A. A. Fröhlich, R. Pellizzoni, and S. Fischmeister, "Implementation and evaluation of global and partitioned scheduling in a real-time OS," *Real Time Syst.*, vol. 49, no. 6, pp. 669–714, 2013.

[14] C. Gu, N. Guan, Q. Deng, and W. Yi, "Partitioned mixed-criticality scheduling on multiprocessor platforms," in *Proc. Design Autom. Test Europe Conf. Exhibit.*, Dresden, Germany, 2014, pp. 1–6.

[15] J.-J. Han, X. Tao, D. Zhu, and H. Aydin, "Criticality-aware partitioning for multicore mixed-criticality systems," in *Proc. 45th Int. Conf. Parallel Process.*, Philadelphia, PA, USA, 2016, pp. 227–235.

[16] O. R. Kelly, H. Aydin, and B. Zhao, "On partitioned scheduling of fixed-priority mixed-criticality task sets," in *Proc. Int. Conf. Trust Security Privacy Comput. Commun.*, 2011, Changsha, China, pp. 1051–1059.

[17] A. Kostrzewa, S. Saidi, and R. Ernst, "Dynamic control for mixed-critical networks-on-chip," in *Proc. 36th IEEE Real Time Syst. Symp.*, San Antonio, TX, USA, 2015, pp. 317–326.

[18] K. Lakshmanan, D. De Niz, R. Rajkumar, and G. Moreno, "Resource allocation in distributed mixed-criticality cyber-physical systems," in *Proc. 30th Int. Conf. Distrib. Comput. Syst.*, Genoa, Italy, 2010, pp. 169–178.

[19] G. Lipari and G. Buttazzo, "Resource reservation for mixed criticality systems," in *Proc. Workshop Real Time Syst. Past Present Future*, York, U.K., 2013, pp. 60–74.

[20] J. M. López, J. L. Díaz, and D. F. García, "Utilization bounds for EDF scheduling on real-time multiprocessor systems," *Real Time Syst.*, vol. 28, no. 1, pp. 39–68, 2004.

[21] M. Neukirchner, P. Axer, T. Michaels, and R. Ernst, "Monitoring of workload arrival functions for mixed-criticality systems," in *Proc. IEEE Real Time Syst. Symp.*, Vancouver, BC, Canada, 2013, pp. 88–96.

[22] 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.

[23] P. Rodriguez, L. George, Y. Abdeddaïm, and J. Goossens, "Multi-criteria evaluation of partitioned EDF-VD for mixed-criticality systems upon identical processors," in *Proc. Workshop Mixed Criticality Syst.*, 2013, pp. 49–54.

[24] E. A. Lester, "Risk-based alternatives to the DO-178C software design assurance process," in *Proc. Digit. Avionics Syst. Conf.*, Prague, Czechia, 2015, pp. 8B2-1–8B2-13.

[25] V. Sciandra, P. Courbin, and L. George, "Application of mixed-criticality scheduling model to intelligent transportation systems architectures," *ACM SIGBED Rev.*, vol. 10, no. 2, p. 22, 2013.

[26] L. Sigrist, G. Giannopoulou, P. Huang, A. Gomez, and L. Thiele, "Mixed-criticality runtime mechanisms and evaluation on multicores," in *Proc. Real Time Embedded Technol. Appl. Symp.*, 2015, Seattle, WA, USA, pp. 194–206.

[27] A. Specification, *651: Design Guidance for Integrated Modular Avionics*, Aeronautical Radio Inc., Annapolis, MD, USA, 1991.

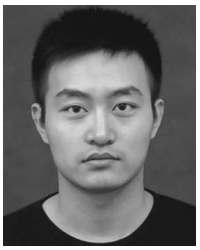
[28] H. Su and D. Zhu, "An elastic mixed-criticality task model and its scheduling algorithm," in *Proc. Conf. Design Autom. Test Europe*, Grenoble, France, 2013, pp. 147–152.

[29] S. Vestal, "Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance," in *Proc. 28th IEEE Real Time Syst. Symp.*, Tucson, AZ, USA, 2007, pp. 239–243.

[30] N. Zhang, C. Xu, J. Li, and M. Peng, "A sufficient response-time analysis for mixed criticality systems with pessimistic period," *J. Comput. Inf. Syst.*, vol. 11, no. 6, pp. 1955–1964, 2015.

Jian-Jun Han (M'07) received the Ph.D. degree in computer science and engineering from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 2005.

He is currently an Associate Professor with the School of Computer Science and Technology, HUST. He was with the University of California at Irvine, Irvine, CA, USA, as a Visiting Scholar from 2008 to 2009, and with the Seoul National University, Seoul, South Korea, from 2009 to 2010. His current research interests include real-time systems, parallel processing, and green computing.



Xin Tao is currently pursuing the master's degree with the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China.

His current research interests include real-time scheduling algorithm, embedded systems, and operating systems.

Dakai Zhu (M'04) received the Ph.D. degree in computer science from the University of Pittsburgh, Pittsburgh, PA, USA, in 2004.

He is currently an Associate Professor with the Department of Computer Science, University of Texas at San Antonio, San Antonio, TX, USA. His current research interests include real-time systems, power aware computing, and fault-tolerant systems.

Dr. Zhu was a recipient of the U.S. National Science Foundation Faculty Early Career Development Award in 2010.

Hakan Aydin (M'02) received the Ph.D. degree in computer science from the University of Pittsburgh, Pittsburgh, PA, USA, in 2001.

He is currently an Associate Professor with the Department of Computer Science, George Mason University, Fairfax, VA, USA. His current research interests include real-time systems, low-power computing, fault tolerance, and real-time operating systems.

Dr. Aydin served as the Technical Program Committee Chair of IEEE RTAS in 2011.

Zili Shao (M'06) received the M.S. and Ph.D. degrees from the Department of Computer Science, University of Texas at Dallas, Richardson, TX, USA, in 2003 and 2005, respectively.

He has been an Associate Professor with the Department of Computing, Hong Kong Polytechnic University, Hong Kong, since 2010. His current research interests include embedded systems, real-time systems, compiler optimization, and hardware/software co-design.

Laurence T. Yang (SM'04) received the Ph.D. degree in computer science from the University of Victoria, Victoria, BC, Canada. He is currently a Professor with the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China, and the Department of Computer Science, St. Francis Xavier University, Antigonish, NS, Canada. His current research interests include parallel and distributed computing, and embedded and ubiquitous/pervasive computing.