IEEE TRANSACTIONS ON
PARALLEL AND
DISTRIBUTED SYSTEMS



# Resource Sharing in Multicore Mixed-Criticality Systems: Utilization Bound and Blocking Overhead

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

**Abstract**—In mixed-criticality (MC) system, diverse application activities with various certification requirements (different criticality) can share a computing platform, where multicore processors have emerged as the prevailing computing engines. Focusing on the problem of resource access contention in multicore MC systems, we analyze the synchronization issues and blocking characteristics of the Multiprocessor Stack Resource Policy (MSRP) with both priority and criticality inversions among MC tasks being considered. We develop the first *criticality-aware utilization bound* under partitioned EDF (Earliest Deadline First) and MSRP by taking the worst case synchronization overheads of tasks into account. The *non-monotonicity* of the bound where it may decrease when more cores are deployed is identified, which can cause anomalies in the feasibility tests. With the objective to improve system schedulability, a novel *criticality-cognizant* and *resource-oriented* analysis approach is further studied to tighten the bound on the synchronization overheads for MC tasks scheduled under partitioned EDF and MSRP. The simulation results show that the new analysis approach can effectively reduce the blocking times for tasks (up to 30%) and thus improve the schedulability ratio (e.g., 10% more). The actual implementation in Linux kernel further shows the *practicability* of partitioned-EDF with MSRP (with run-time overhead being about 3% to 7% of the overall execution time) for MC tasks running on multicores with shared resources.

**Index Terms**—Mixed-criticality systems; Multicore; Shared resources; Resource access contention; Utilization bound; Run-time overhead;

## 1 Introduction

Modern complex embedded systems need to integrate multiple functionalities into a shared computing platform due to space, power and cost concerns. For instance, the IMA (Integrated Modular Avionics) initiative for aerospace applications aims at hosting multiple avionics components on a shared system. In such integrated systems, activities with various certification requirements (different levels of *criticality*) can co-exist. Considering the avionics certification standard DO-178C [1] as an example, it defines five design assurance levels (A-E) that are categorized by the degree of hazards arisen from run-time failures. To incorporate such certification requirements, the concept of *Mixed-Criticality (MC)* systems was proposed in [2] and the Mixed-Criticality Scheduling (MCS) has been studied extensively for various system and task patterns over the past decade [3], [4], [5], [6], [7], [8].

For the co-running tasks on a common platform, a task may need to exclusively access a shared resource (e.g., data objects) to protect its integrity. However, the synchronization requirements due to resource access contention can cause *priority inversion* [9], [10]. When a low-priority task exclusively accesses a resource, a high-priority task will be blocked when it requests access to the same resource where it has to wait until the low-priority task completes using the resource. Such blocking caused by shared resource access can lead to additional delays for high-priority tasks and thus degraded system schedulability.

To tackle such problem of *task synchronization*, several lock based resource access protocols have been investigated. For single processor systems, the most notable protocols are Priority Ceiling Protocol (PCP) [10] under fixed-priority scheduling and Stack Resource Policy (SRP) [9] under dynamic-priority scheduling. With the emergence of multiprocessor/multicore, there is a renascent research interest in the resource access synchronization problem since such problem becomes more salient and requires more effective solutions. Specifically, an empirical study pointed out that the contention for data structures would increase the amount of an application's execution time up to about 30 percent on a system with sixteen cores due to busy-waiting for shared resources [11]. Following the same principle, the above protocols designed for uniprocessors have extended for multiprocessors, for instance, MSRP (Multiprocessor SRP) [12], MPCP (Multiprocessor PCP) [13] and DPCP (Distributed PCP) [14]. Moreover, spin locks have been widely applied in multiprocessor real-time systems [15], e.g., AUTOSAR (AUTomotive Open System ARchitecture) standard [16]. An empirical work further shows that when compared to suspension scheme (e.g., [13], [14], [17]), the *spin lock* based scheme (e.g., [12], [15]) can effectively improve system schedulability in the context of multiprocessors [18].

Although extensive work has been done to handle the resource access contention problem, the work on the task synchronization in MC systems is quite limited. Here, in addition to priority inversion, tasks in MC systems can experience *criticality inversion* [19], [20]: if a low-criticality task is holding a resource when the system switches to a higher criticality level, the task cannot be dropped immediately and will block the executions of high-criticality tasks. Such criticality inversion can significantly complicate the problem of resource access contention and consequently affects the schedulability analysis for MC systems.

To address the problem of task synchronization for MC systems with single processor, several recent studies on the resource access protocols were reported in [19], [20], [21], [22]. Generally, the principle of these protocols is to avoid unbounded criticality inversions (in addition to priority inversions) caused by medium-criticality tasks. Some recent studies have focused on the communication contention for multicore MC systems [5], [6]. However, this paper targets at shared data (e.g., global variables) among

- *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. Email:jasonhan@hust.edu.cn, m201472823@hust.edu.cn, ltyang@stfx.ca. Laurence T. Yang is the corresponding author.*
- *D. Zhu is with the Department of Computer Science, The University of Texas at San Antonio, San Antonio, TX 78249. Email:dzhu@cs.utsa.edu*

Digital Object Identifier 10.1109/TPDS.2017.2677442

tasks [21], [22], [23], instead of hardware resources (e.g., network and storage) that can be handled with their corresponding schemes.

Although EDF-VD (Earliest-Deadline-First with Virtual Deadlines) [24] was shown to be an effective scheduler for MC systems when no shared resource is considered, *additional* priority inversions can occur under EDF-VD with SRP in uniprocessors [21]. This comes from the scenario of *preempt-then-block*: for a *dual-criticality system* where there are only two criticality levels, a high-criticality task can be preempted by another high-criticality one having a smaller *virtual* deadline; while when the system switches to high-criticality mode where the original deadlines of high-criticality tasks are restored, the preempted task can be blocked again if another high-criticality one has a larger *real* deadline.

More importantly, additional global waiting times (as a portion of blocking times) for tasks accessing shared resources in multicores need to be considered [12], [15], [23]. Therefore, with the synchronization overheads being considered in multicores, the schedulability advantages of EDF-VD may be intuitively offset when compared to EDF, especially for the scenarios where tasks need to access and/or wait for resources for a long time.

To the best of our knowledge, there is no existing work on the scheduling of MC tasks with multiple criticality that access shared resources (e.g., data objects) in multicores. With the objective to address this problem and reduce the impacts of resource access contention on the tasks' schedulability, we focus on EDF (*instead of EDF-VD* which is left to our future work) as the base scheduler for MC tasks under MSRP resource access protocol [12]. The **contributions** of this work can be summarized as follows:

- We characterize the blocking issues of MSRP for MC tasks running on multicores with shared resources: a task can be blocked **at most once** due to priority inversion and **at most once** due to criticality inversion at each *valid* level;
- We develop the *first* **criticality-aware utilization bound** for MC tasks accessing resources under partitioned-EDF and MSRP with WFD (Worst-Fit Decreasing) mapping;
- We identify the **non-monotonicity** of the bound, where the bound can decrease with more deployed cores due to the increased inter-core synchronization interference;
- We further explore a new analysis approach to **tightening the upper-bound** on the synchronization overheads of MC tasks scheduled by partitioned-EDF with MSRP.
- The empirical results from **actual implementation** in Linux show that the partitioned EDF scheduler with MSRP is practically viable due to its acceptable run-time overhead.

The remainder of this paper is organized as follows. The related work is briefly reviewed in Section 2. Section 3 presents system models and preliminaries of MSRP. The blocking issues for MC tasks under MSRP are analyzed in Section 4. The criticality-aware utilization bound under partitioned-EDF with MSRP and its non-monotonicity are discussed in Section 5. A new analysis approach to reducing the bound of synchronization overheads for MC tasks is given in Section 6. The implementation and evaluation results are discussed in Section 7 and Section 8 concludes the paper.

## 2 Closely Related Work

To address the task synchronization problem, several lock-based resource access protocols have been developed. For uniprocessor systems, Sha *et al.* proposed a protocol PCP [10] under fixed-priority scheduling (e.g., RMS (Rate-Monotonic Scheduling)) and Baker investigated a protocol SRP [9] under EDF.

Based on the same principle, the above protocols applied in single processor systems have been extended to multiprocessor systems, e.g., MSRP [12], MPCP [13], etc. The Flexible Multiprocessor Locking Protocol (FMLP) [11] and a suspension-based optimal locking protocol (OMLP) [25] were also studied. Recently, the Multiprocessor resource sharing Protocol (MrsP) and its feasibility analysis were developed for partitioned fixed-priority scheduling [26]. Several recent studies on the spin lock mechanism can be found in [15], [23]. In addition, under the constraints of task synchronization due to shared resource access, several task partitioning algorithms were reported in [27], [28], [29] and a few energy management schemes were reported in [27], [30].

While extensive studies have been done to tackle the resource access contention problem, the work on the task synchronization in MC systems is quite limited. To deal with such problem for fixed priority MC scheduling in uniprocessors, Burns exploited PCP and incorporated blocking items due to shared resource access into the response time analysis [22]. Lakshmanan *et al.* extended zero slack scheduling to handle task synchronization and proposed two protocols: Priority and Criticality Inheritance Protocol (PCIP) and Priority and Criticality Ceiling Protocol (PCCP) [19]. Zhao *et al.* proposed a Highest-Locker Criticality Priority Ceiling Protocol (HLC-PCP) [20]. More recently, based on EDF-VD [24], Zhao *et al.* [21] integrated SRP and the preemption threshold scheduling [31], and studied a Mixed-Criticality Stack Resource Policy (MC-SRP) for dual-criticality systems with single processor.

For multiprocessor MC systems, a resource access scheme, where all shared resources are placed in designated servers and any task requesting resources needs to contact the servers via a MC Inter-Process Communication (MC-IPC) protocol, was reported in [32]. Nonetheless, this scheme may not be appropriate for shared-resource multicore systems, where the support of the protocol and servers for such centralized resources have to be developed [33]. A comprehensive review of mixed-criticality scheduling can be found in [33].

## 3 System Models and Preliminaries

In this section, we first present the system and task models. The MSRP protocol is briefly reviewed, followed by the introduction of the schedulability condition for MC tasks running on multicores with shared resources under partitioned-EDF and MSRP.

### 3.1 System and Task Models

We consider a homogeneous multicore processor that consists of $M$ processor cores ($\{\mathcal{P}_1, \ldots, \mathcal{P}_M\}$) having identical capabilities and functions. A set of $N$ *implicit-deadline* MC tasks $\Psi = \{\tau_1, \ldots, \tau_N\}$ run on the system with their initial arrival being time 0. The subset of tasks assigned to core $\mathcal{P}_m$ is denoted as $\Psi_m$ and there is $\Psi = \cup_{m=1}^{M} \Psi_m$. There are $K$ ($> 1$) criticality levels for the tasks, where the system starts its operation at level-1.

Every MC task $\tau_i$ is characterized by a tuple of parameters: $\tau_i = \{\ell_i\, p_i, \mathbf{C}_i\}$. Here, $\ell_i$ ($\leq K$) denotes task $\tau_i$'s criticality level (i.e., its *own* criticality). With the focus on implicit-deadlines, $p_i$ refers to task $\tau_i$'s period as well as its relative deadline. The vector $\mathbf{C}_i =< c_i(1), \ldots, c_i(\ell_i) >$ represents the worst case execution times (WCETs) of task $\tau_i$ at each *valid* level: its WCET at higher level is usually larger than that at lower level, i.e., $c_i(1) < \ldots < c_i(\ell_i)$. Following the common assumption for MC task system [2], any task $\tau_i$ executes for no more than its maximum WCET $c_i(\ell_i)$ at run-time. Moreover, since there is only one active job (instance)

for each implicit-deadline task at any time, we use task and job *interchangeably* in this paper unless otherwise specified.

Note that, in the context of MC systems, it is important that the low-criticality tasks do not interfere with high-criticality tasks for temporal and logic isolation between criticality levels, whereas it is not clear to what extent data should flow across criticality [33]. Considering the shared memory multicores, we assume that every resource can be shared by all tasks. However, our proposed **general approaches** in this work can be easily altered to address the *special case*, where only tasks of the same criticality can share resources to promote isolation between criticality [21], [22], [32].

The system has $R$ global resources $\mathcal{R} = \{\mathcal{R}_1, \ldots, \mathcal{R}_R\}$ that can be shared by all tasks and are protected within critical sections. At any time, a resource can be held by only one task within one of its critical sections, i.e., the access to any resource by tasks is *exclusive*. Moreover, non-nested lock accesses are the common case in practice and the nested critical sections can be transformed to non-nested ones with group lock [11], [15]. Therefore, we only consider *non-nested critical sections* here: at any time, a task is not allowed to request a resource while holding another one.

Note that, a task may need to access a resource multiple times within its different critical sections. There are $n_i$ critical sections in task $\tau_i$ and the $x^{th}$ critical section ($x = 1, \ldots, n_i$) is denoted as $z_{i,x}$ that accesses resource $r_{i,x}$ ($\in \mathcal{R}$). Different from the assumption where each critical section has varying execution time estimates at different levels [22], here the largest size of any critical section $z_{i,x}$ is assumed to be fixed [19], [21] and it is denoted as $c_{i,x}$.

Moreover, we adopt a *global* variant of adaptive mixed criticality (*AMC*) scheme [34], [35] in this work, where the system has the ability to monitor the executions of individual jobs [7]. When a task $\tau_i$ ($\ell_i > k$) executes for more than its level-$k$ WCET, the system performs a global mode transition and switches to level-$(k+1)$: *all* tasks in the system with their own criticality level being $k$ will be dropped (after exiting *all* their critical sections [19], [20], if applicable), and level-$k$ tasks cannot be released until the system becomes idle and returns to level-1 execution mode.

### 3.2 Resource Access Protocol MSRP

As one of the notable resource access protocols for multiprocessors, the protocol MSRP considered in this work can effectively tackle task synchronization for multiprocessor applications, e.g., power-train controller for automotive activities [12] and GAP (Generic Avionics Platform) task set for avionics activities [36]. First, for *non-MC task system* (i.e., conventional real-time task set without criticality certification), we briefly review the principles of MSRP with its basic rules being summarized as follows:

- **Rule 1:** On each processor core, tasks are scheduled by the EDF scheduler. When a task $\tau_i$ issues a request for a resource $\mathcal{R}_a$, if $\mathcal{R}_a$ is free, it will lock and access resource $\mathcal{R}_a$ non-preemptively; otherwise, if $\mathcal{R}_a$ is currently held by another task on a different core, task $\tau_i$ will be added to the FIFO queue of resource $\mathcal{R}_a$ and busy-waits for it;
- **Rule 2:** Once task $\tau_i$ finishes accessing a resource $\mathcal{R}_a$, it releases $\mathcal{R}_a$ and becomes preemptable again. If $\mathcal{R}_a$'s FIFO queue is not empty (i.e., there are tasks from other cores waiting for accessing $\mathcal{R}_a$), the header task is de-queued and starts accessing the resource; otherwise, $\mathcal{R}_a$ is unlocked.

Note that, a task needs to start spinning until it gains access to the resource required subject to the spin lock mechanism [12], [15], [23]. From the above steps, we can see that the execution of a non-MC task $\tau_i$ on core $\mathcal{P}_m$ can be blocked due to resource access synchronization at two different occasions as follows:

- When task $\tau_i$ attempts to access a resource $\mathcal{R}_a$ that is currently occupied by a task from another core, it has to wait in the FIFO queue of resource $\mathcal{R}_a$ and such duration caused by synchronization interference from other cores is denoted as *global waiting time*;
- When a low-priority task on core $\mathcal{P}_m$ is holding or busy-waiting for a resource that is currently held by a task on a different core, task $\tau_i$ can be blocked and such duration is denoted as *local blocking time* (due to priority inversion).

For *non-MC tasks* accessing resources in *non-nested* critical sections under MSRP, we can have the following properties [12]:

**Property 1.** *For any processor core at any given time, there exists* at most one *task that is either holding or busy-waiting for a resource that is currently occupied by a task on another core.*

**Property 2.** *A task can be blocked (due to priority inversion) by a low-priority task on the same core* at most once.

**Property 3.** *A task's local blocking time is upper bounded by the* longest duration *of any low-priority task on the same core that accesses (and waits for, if possible) one of its resources* once.

We first review the schedulability condition for *non-MC tasks* under partitioned EDF and MSRP. For the convenience of presentation, some notations are first defined as follows:

- $BW_{i,x}$: denotes the maximum amount of time that task $\tau_i$ waits for accessing resource $r_{i,x}$ in its critical section $z_{i,x}$;
- $BW_i$: indicates the worst-case global waiting time taken by task $\tau_i$ to access all its resources, i.e., $BW_i = \sum_{x=1}^{n_i} BW_{i,x}$;
- $B_i$: refers to the maximum local blocking time for task $\tau_i$.

Then, the feasibility condition for non-MC tasks under partitioned EDF and MSRP can be summarized as follows [9], [12].

**Theorem 1.** *For a given mapping of tasks to cores, the* non-MC tasks *running on a multicore system with* shared resources *are schedulable under partitioned-EDF and MSRP if, for every processor core $\mathcal{P}_m$ ($m = 1, \ldots, M$), there is:*

$$\forall \tau_i \in \Psi_m, \frac{B_i}{p_i} + \sum_{\forall \tau_j \in \Psi_m}^{p_j \le p_i} \frac{c_j + BW_j}{p_j} \le 1 \tag{1}$$

where for non-MC task $\tau_i$, $c_i$ denotes its traditional WCET and $B_i$ *only* accounts for the local blocking time due to priority inversion.

### 3.3 Schedulability Condition for Partitioned-EDF and M-SRP in MC Systems

We next review the schedulability condition for MC tasks under partitioned-EDF when *no shared resource* is considered. A comprehensive review of static scheduling algorithms for multiprocessor systems can be found in [37].

Here, the *level-k utilization* of task $\tau_i$ ($k \le \ell_i$) is defined as $u_i(k) = \frac{c_i(k)}{p_i}$, and the *total level-k utilization* of tasks on core $\mathcal{P}_m$ with their own criticality level being $j$ ($\ge k$) is defined as:

$$U_j^{\Psi_m}(k) = \sum_{\forall \tau_i : \tau_i \in \Psi_m}^{\ell_i = j \wedge j \ge k} u_i(k). \tag{2}$$

For a set $\Psi$ of MC tasks with $K$ criticality levels running on a system with $M$ cores under the partitioned-EDF scheduler, the *sufficient* schedulability condition for every core $\mathcal{P}_m$ ($m = 1, \ldots, M$) can be given as follow (see Theorem 3.4 in [24]):

$$\sum_{k=1}^{K} U_k^{\Psi_m}(k) \leq 1. \tag{3}$$

Basically, the equation says that if core $\mathcal{P}_m$ can accommodate the *maximum* utilization demands of all its tasks at their own criticality levels, the tasks on $\mathcal{P}_m$ are feasible under the EDF scheduler.

Finally, following the schedulability condition for MC tasks under partitioned EDF as given in Equation (3) and the same reasoning of the feasibility results for non-MC tasks under partitioned EDF and MSRP as given in Equation (1), we can obtain:

**Proposition 1.** *For a set* $\Psi$ *of* MC tasks *with K criticality levels that access* shared resources *in a multicore system with M cores, the task set is feasible under the partitioned-EDF scheduler with MSRP if, for every core* $\mathcal{P}_m$ *(*$m = 1, \ldots, M$*), there is:*

$$\forall \tau_i \in \Psi_m, \frac{B_i}{p_i} + \sum_{\forall \tau_j \in \Psi_m}^{p_j \leq p_i} \frac{c_j(\ell_j) + BW_j}{p_j} \leq 1. \tag{4}$$

Essentially, the equation indicates that by incorporating the maximum utilization demands of MC tasks, when Equation (1) holds for any core $\mathcal{P}_m$ ($m = 1, \ldots, M$), the task set $\Psi$ is guaranteed to be feasible under partitioned-EDF with MSRP. Here, $B_i$ represents the maximum local blocking time experienced by task $\tau_i$ due to *both priority and criticality inversions*.

## 4 MSRP in Multicore Mixed-Criticality Systems

The resource access protocol MSRP does not consider MC systems where criticality inversion can occur. Basically, the principle of a resource access protocol for MC systems is to ensure that the blocking time for a task due to priority and criticality inversions should be as little as possible to reduce the tasks' synchronization overheads for better system schedulability [19], [20], [21], [22].

Fortunately, the spin lock mechanism exploited by MSRP can prevent MC tasks suffering from not only unbounded priority inversions [12] but unbounded criticality inversions at each level. In what follows, we discuss the detailed synchronization issues for MC tasks that access shared resources under the partitioned-EDF scheduler with MSRP.

### 4.1 The Blocking Characteristics of MSRP in MC Systems

When the system operates at criticality level-$k$ and no task in the system executes for more than its level-$k$ WCET, we say that the system is in *normal mode* at level-$k$.

With the focus on the spin lock mechanism in MSRP, a task cannot be dropped immediately when it is holding or busy-waiting for a resource, and it can be *safely* discarded only after exiting all its critical sections [15], [19], [20]. Therefore, once a task $\tau_i$ ($\ell_i > k$) runs for more than its level-$k$ WCET (i.e., $c_i(k)$) without indicating its completion, the system is in *mode-change period* at level-$k$, which is defined as the time interval that starts when task $\tau_i$ exceeds its $c_i(k)$ until all level-$k$ tasks are dropped upon exiting *all* their critical sections (if applicable). After such operations, the system switches to level-$(k+1)$ execution mode.

Hence, for MC tasks scheduled under partitioned EDF and MSRP, blocking occurs if any of the conditions below is satisfied:

- A high-priority task is blocked by a low-priority task on the same core that is busy-waiting for (if possible) or accessing a resource, which is called priority-inversion blocking and abbreviated as *pi-blocking*;
- During mode-change period at any level-$k$ ($< K$), a high-criticality task is blocked when a low-criticality task with its own criticality level being $k$ or lower (which is discussed in the proof of Property 6) on the same core has not exited all its critical sections, which is called criticality-inversion blocking and abbreviated as *ci-blocking*.

For any task $\tau_i$, we can classify tasks other than $\tau_i$ into four categories: a) $hpH(\tau_i)$: the subset of tasks with both higher priority and higher criticality level than task $\tau_i$, i.e., for each task $\tau_j$ ($\in hpH(\tau_i)$), there are $p_j < p_i$ [9], [12] and $\ell_j > \ell_i$; b) $hpL(\tau_i)$: the subset of tasks with higher priority but lower criticality level than task $\tau_i$; c) $lpH(\tau_i)$: the subset of tasks with higher criticality level but lower priority than task $\tau_i$; and d) $lpL(\tau_i)$: the subset of tasks with both lower priority and lower criticality level than task $\tau_i$.

We can see that any task in the subset $hpH(\tau_i)$ cannot block the execution of task $\tau_i$, any task in in the subset $hpL(\tau_i)$ can cause only ci-blocking for task $\tau_i$ during mode-change period, any task in in the subset $lpH(\tau_i)$ can lead to only pi-blocking for task $\tau_i$, and any task in in the subset $lpL(\tau_i)$ can result in both pi-blocking and ci-blocking for task $\tau_i$.

Obviously, Property 1 still holds for MSRP in MC systems due to its inherent spin lock mechanism. Also, by exploiting the spin lock mechanism, Property 2 holds for MC tasks in the set $\Psi$ scheduled by partitioned-EDF and MSRP as follows.

**Property 4.** *Any MC task that accesses shared resources in a multicore system can experience pi-blocking* at most once *under the partitioned-EDF scheduling with MSRP.*

**Property 5.** *No MC task can be blocked due to pi-blocking after it starts execution under partitioned-EDF and MSRP.*

However, Property 3 fails due to additional ci-blocking arisen in MC systems. As for the ci-blocking under partitioned-EDF and MSRP, we can obtain the property as follows.

**Property 6.** *For any MC task* $\tau_i$ *that accesses shared resources in a multicore system, it can experience ci-blocking* at most once *at any level-k (*$k = 1, \ldots, \ell_i - 1$*) under partitioned-EDF and MSRP.*

*Proof.* Suppose that the system operates at level-$k$ mode and a high criticality task $\tau_i$ ($\ell_i > k$) is released on core $\mathcal{P}_m$. When the mode transition occurs (i.e., a task runs for more than its level-$k$ WCET), we assume that task $\tau_i$ has not completed its execution and a low criticality task $\tau_j$ ($\ell_j \geq k \wedge \ell_j < \ell_i$) on $\mathcal{P}_m$ is busy-waiting for or holding a resource $\mathcal{R}_a$ in its critical section $z_{j,y}$. Here, task $\tau_j$ cannot be considered as blocking task $\tau_i$ due to ci-blocking if task $\tau_j$'s own criticality level is higher than $k$ (i.e., $\ell_j > k$).

Otherwise, when there is $\ell_j = k$, $\tau_i$ is blocked by $\tau_j$ due to ci-blocking. For such case, $\tau_j$ continues accessing $\mathcal{R}_a$ due to the spin lock mechanism of MSRP, and it is immediately dropped upon exiting its critical section $z_{j,y}$. After that, based on Property 1 and the AMC scheme, there is no level-$k$ task on core $\mathcal{P}_m$ any more, until the system becomes idle and gets back to level-1 mode.

Note that, before task $\tau_j$ finishes using resource $\mathcal{R}_a$, the system can switch to a higher level (i.e., $k$+2) once a task exceeds its level-$(k+1)$ WCET. For such case, when there is $\ell_j = k$, the property still holds based on Property 1 as there is only one task (i.e., $\tau_j$) that is executing (in its critical section) on core $\mathcal{P}_m$ now: either $\tau_i$

is blocked by $\tau_j$ during mode-change periods at level-$k$ and level-$(k+1)$ if $\ell_i > k+1$, or $\tau_i$ itself is dropped if $\ell_i = k+1$ (here $\tau_i$ experiences ci-blocking during mode-change period at level-$k$). Also, once $\tau_j$ exits critical section $z_{j,y}$, there is no level-$(k+1)$ task on core $\mathcal{P}_m$ any more until the system returns to level-1 mode. □

Therefore, based on the above analysis, we can directly obtain the properties related to the blocking characteristics for MC tasks scheduled under partitioned-EDF and MSRP as follows.

**Property 7.** *For any MC task $\tau_i$ that accesses shared resources in a multicore system, it can experience ci-blocking* at most $(\ell_i - 1)$ *times under partitioned-EDF and MSRP.*

**Property 8.** *Whichever blocking (i.e., pi-blocking or ci-blocking) happens, any job of a MC task can be blocked for* at most one *critical section by a job of other tasks on the same core under partitioned-EDF and MSRP.*

In summary, by exploiting spin lock mechanism, the resource access protocol MSRP can avoid both unbounded pi-blocking and unbounded ci-blocking for MC tasks scheduled by partitioned EDF that access shared resources in multicore systems.

### 4.2 Synchronization Overheads of MC Tasks under MSRP

Now we formalize the bound on the synchronization overheads of MC tasks scheduled by partitioned-EDF with MSRP.

For a given task-to-core partition, the maximum global waiting time $BW_{i,x}$ taken by task $\tau_i$ on core $\mathcal{P}_m$ to access $\mathcal{R}_a$ in its critical section $z_{i,x}$ ($r_{i,x} = \mathcal{R}_a$) under MSRP can be calculated as [12]:

$$BW_{i,x} = \sum_{j=1,\cdots,M}^{j\neq m} tp_j^{max}(\mathcal{R}_a) \tag{5}$$

where $tp_j^{max}(\mathcal{R}_a)$ is the longest time of any task from another core $\mathcal{P}_j$ ($j \neq m$) spent in accessing resource $\mathcal{R}_a$ *once*. That is, in the worst case scenarios, task $\tau_i$ may have to wait for the longest access time of resource $\mathcal{R}_a$ by tasks on all other cores, and thus $tp_j^{max}(\mathcal{R}_a)$ can be further computed as:

$$\begin{aligned} tp_j^{max}(\mathcal{R}_a) &= \max\{tt_l^{max}(\mathcal{R}_a) | \forall \tau_l \in \Psi_j\} \\ tt_l^{max}(\mathcal{R}_a) &= \max\{c_{l,y} | \forall z_{l,y} : r_{l,y} = \mathcal{R}_a\} \end{aligned} \tag{6}$$

where $tt_l^{max}(\mathcal{R}_a)$ denotes the maximum amount of time for task $\tau_l$ ($\in \Psi_j$) to access resource $\mathcal{R}_a$ *once*.

Then, based on $BW_{i,x}$, the worst case total global waiting time $BW_i$ taken by task $\tau_i$ to access all its resources required can be simply accumulated as $BW_i = \sum_{x=1}^{n_i} BW_{i,x}$.

Note that, the local blocking time $B_i$ of task $\tau_i$ is comprised of two parts: local pi-blocking time $B_i^{pi}$ and local ci-blocking time $B_i^{ci}$. Based on Property 4, the maximum local pi-blocking time experienced by task $\tau_i$ can be found as [12]:

$$B_i^{pi} = \max\{BW_{j,y} + c_{j,y} | \forall z_{j,y} : \tau_j \in \Psi_m \wedge p_j > p_i\}. \tag{7}$$

We denote the maximum local ci-blocking time for task $\tau_i$ during mode-change period at level-$k$ ($< \ell_i$) as $B_i^{ci}(k)$. Based on Property 6, it can be calculated as:

$$B_i^{ci}(k) = \max\{BW_{j,y} + c_{j,y} | \forall z_{j,y} : \tau_j \in \Psi_m \wedge \ell_j = k\}. \tag{8}$$

Then, the maximum local ci-blocking time experienced by task $\tau_i$ can be simply accumulated as $B_i^{ci} = \sum_{k=1}^{\ell_i - 1} B_i^{ci}(k)$.

Finally, the worst-case local blocking time $B_i$ for task $\tau_i$ due to both pi-blocking and ci-blocking can be found as $B_i = B_i^{pi} + B_i^{ci}$.

## 5 Criticality-Aware System Utilization Bound

For real-time task systems without shared resource access, López *et al.* have studied the utilization bounds for partitioned-EDF with various mapping heuristics, such as WFD [38]. As long as the total utilization of a task set does not exceed such bounds, the partitions generated by these heuristics are guaranteed to be schedulable, which can be utilized for efficient feasibility test. Specifically, such bounds increase *monotonically* with more available processors [38]. Below, we study a criticality-aware utilization bound for MC tasks under partitioned-EDF and MSRP with its non-monotonicity being identified.

### 5.1 Criticality-Aware Utilization Bound with MSRP

We can see that the schedulability condition given in Equation (3) considers the tasks' maximum utilizations at their *own criticality*, which is quite different from the feasibility results for *non-MC* task system where no criticality certification needs to be considered.

Since each MC task has *basic* level-1 WCET, we first convert the sufficient condition presented in Equation (3) to the *simplified* schedulability condition for the total level-1 utilization of tasks on any core (i.e., *level*-1 *core utilization limit*). Then, based on such limit, we study a criticality-aware utilization bound for MC tasks accessing shared resources under partitioned-EDF and MSRP.

Define $\omega$ as the maximum ratio of WCETs between consecutive criticality levels for any task, that is, $\omega = \max_{\forall \tau_i}\{\frac{c_i(k+1)}{c_i(k)} | k = 1, \ldots, \ell_i - 1\}$ where $\omega > 1$. Based on the definition of $\omega$, we can obtain the following theorem regarding to the level-1 core utilization limit for any core when no shared resource is considered.

**Theorem 2.** Without the consideration of shared resource access, *for a set $\Psi_m$ of MC tasks assigned to core $\mathcal{P}_m$, the tasks are feasible under EDF if, the total level-*1 *utilization of tasks satisfies:*

$$\sum_{j=1}^{K} U_j^{\Psi_m}(1) \leq \theta = \frac{1}{\frac{\omega^K - 1}{\omega - 1} - (K-1)}. \tag{9}$$

*Proof.* We conduct the proof by contrapositive. Based on Equation (2) and the definition of $\omega$, we can get $U_k^{\Psi_m}(k) \leq \omega^{k-1} \cdot U_k^{\Psi_m}(1)$ ($k = 2, \ldots, K$) and $\sum_{k=1}^{K} U_k^{\Psi_m}(k) \leq U_1^{\Psi_m}(1) + \omega \cdot U_2^{\Psi_m}(1) + \ldots + \omega^{K-1} \cdot U_K^{\Psi_m}(1)$. If the sufficient schedulability condition in Equation (3) fails for core $\mathcal{P}_m$ (i.e., $\sum_{k=1}^{K} U_k^{\Psi_m}(k) > 1$), we can obtain:

$$\sum_{j=1}^{K} U_j^{\Psi_m}(1) + \left[(\omega - 1) \cdot U_2^{\Psi_m}(1) + \ldots + (\omega^{K-1} - 1) \cdot U_K^{\Psi_m}(1)\right] > 1.$$

Since $U_x^{\Psi_m}(1) \leq \sum_{j=1}^{K} U_j^{\Psi_m}(1)$ ($x = 1, \ldots, K$), we can further have:

$$\begin{aligned} &\sum_{j=1}^{K} U_j^{\Psi_m}(1) + (\omega - 1 + \ldots + \omega^{K-1} - 1) \cdot \sum_{j=1}^{K} U_j^{\Psi_m}(1) > 1 \\ \Rightarrow &\sum_{j=1}^{K} U_j^{\Psi_m}(1) > \frac{1}{1 + \sum_{k=1}^{K-1}(\omega^k - 1)} = \frac{1}{\frac{\omega^K - 1}{\omega - 1} - (K-1)} = \theta. \end{aligned} \tag{10}$$

Taking its contrapositive, we have $\sum_{k=1}^{K} U_k^{\Psi_m}(k) \leq 1$ when there is $\sum_{j=1}^{K} U_j^{\Psi_m}(1) \leq \theta$, which concludes the proof. □

**Property 9.** *The level-*1 *core utilization limit $\theta$ for any core $\mathcal{P}_m$ ($m = 1, \ldots, M$) decreases when $K$ or $\omega$ increases.*

*Proof.* Let $f(K) = 1 + \sum_{k=1}^{K-1}(\omega^k - 1)$. From Equation (10), there is $\theta = \frac{1}{f(K)}$ . We can see that $f(K)$ increases when $K$ increases and $\omega$

is fixed. Re-define $f(\omega) = 1 + \sum_{k=1}^{K-1}(\omega^k - 1)$. Thus, $f(\omega)$ increases when $\omega$ increases and $K$ is fixed, which concludes the proof. □

Based on the level-1 core utilization limit $\theta$ for any core and the same reasoning of feasibility results for SRP under EDF (see Theorem 10 in [9]), we can obtain the proposition as follows.

**Proposition 2.** *For a set $\Psi$ of MC tasks accessing* shared resources *on an M-core system, the task set is schedulable under partitioned EDF and MSRP if, for every core $\mathcal{P}_m$ ($m = 1, \ldots, M$), there is:*

$$\forall \tau_i \in \Psi_m, \frac{B_i}{p_i} + \sum_{\forall \tau_j \in \Psi_m}^{p_j \le p_i} \frac{c_j(1) + BW_j}{p_j} \le \theta \tag{11}$$

where the maximum synchronization overheads of MC tasks are incorporated into the level-1 core utilization limit. Therefore, when compared to Equation (4), Equation (11) stands for a much pessimistic sufficient condition regarding to tasks' schedulability.

From the discussions in Section 4.2, a task's synchronization overhead rather depends on a specific partition of tasks to cores. To find the upper-bounds for such overheads and then obtain the utilization bound, we first define some notations as follows:

- $p^{min}$: the minimum period of tasks under consideration;
- $n^{max,cs}$: the maximum number of critical sections for a task;
- $c^{max,cs}$: the largest size of critical sections among all tasks;
- $BW^{ub}$: the upper-bound on the overall global waiting time taken by any task to access all its required resources. Based on Equation (5), we can get $BW^{ub} = n^{max,cs} \cdot (M-1) \cdot c^{max,cs}$;
- $B^{ub}$: the upper-bound on the local blocking time (due to priority and criticality inversions) that can be experienced by a task; Based on Properties 4 and 6, we can safely have $B^{ub} = K \cdot (c^{max,cs} + (M-1) \cdot c^{max,cs}) = K \cdot M \cdot c^{max,cs}$;
- $\alpha$: the maximum synchronization-aware level-1 utilization of tasks, which is defined as $\alpha = \max\{\frac{c_i(1)+BW^{ub}}{p_i} | \forall \tau_i \in \Psi\}$;
- $\beta$: the minimum number of MC tasks that can feasibly fit into one core under EDF and MSRP when taking the worst case synchronization overheads into account;
- $\gamma$: the upper-bound for additional utilization on any core due to local blocking time, which is defined as $\gamma = \frac{B^{ub}}{p^{min}}$;
- $\sigma$: the criticality-cognizant synchronization overhead factor, which is defined as $\max\{\frac{B^{ub}}{p^{min}}, \frac{BW^{ub}}{p^{min}}\} = \max\{\gamma, \frac{BW^{ub}}{p^{min}}\}$.

Based on Proposition 2, by incorporating $BW^{up}$ and $B^{up}$ into Equation (11), we can obtain the lemma related to the feasibility condition for MC tasks under partitioned-EDF and MSRP below.

**Lemma 1.** *For a set $\Psi$ of MC tasks that access* shared resources *in a multicore system with M cores, a task-to-core partition is schedulable under partitioned-EDF with MSRP if, for every core $\mathcal{P}_m$ ($m = 1, \ldots, M$), there is:*

$$\forall \tau_i \in \Psi_m, \frac{B^{ub}}{p^{min}} + \sum_{\forall \tau_j \in \Psi_m}^{p_j \le p_i} \frac{c_j(1) + BW^{ub}}{p_j} \le \theta. \tag{12}$$

Based on Lemma 1 and the definitions of $\alpha$ and $\gamma$, we can have $\beta = \lfloor \frac{\theta-\gamma}{\alpha} \rfloor$. Then, following the similar reasoning as in [38], we can easily obtain the lemma with regard to the number of tasks and system schedulability as follows.

**Lemma 2.** *For a set of N implicit-deadline MC tasks running on an M-core system with shared resources, the task set is guaranteed to be schedulable under partitioned-EDF and MSRP if $N \le \beta \cdot M$.*

Finally, based on Equation (12), we can obtain the following theorem with respect to the *criticality-aware level-1 utilization bound ($U^{ca,bound}$)* for MC tasks that access shared resources under partitioned-EDF and MSRP with the WFD heuristic.

**Theorem 3.** *For a set of N implicit-deadline MC tasks with K criticality levels that access shared resources in a multicore system with M cores where the number of tasks $N > \beta \cdot M$, the criticality-aware level-1 utilization bound $U^{ca,bound}$ for partitioned-EDF and MSRP with the WFD mapping heuristic can be found as:*

$$U^{ca,bound} = \min\{U^{b1}, U^{b2}\}, \tag{13}$$

$$U^{b1} = \frac{\beta \cdot M + 1}{1 + \beta} \cdot (\theta - \sigma) - (\beta \cdot M + 1) \cdot \sigma, \tag{14}$$

$$U^{b2} = \frac{M \cdot N}{M + N - 1} \cdot (\theta - \sigma) - N \cdot \sigma. \tag{15}$$

*Proof.* With the focus on the WFD mapping, we assume that tasks have been ordered by their non-ascending level-1 utilizations. Suppose task $\tau_n$ is the first task that fails the feasibility condition for each core as given in Equation (12) when it is assigned to any core. Then, for each core $\mathcal{P}_m$ ($m = 1, \cdots, M$), we can obtain:

$$\frac{B^{ub}}{p^{min}} + \frac{c_n(1) + BW^{ub}}{p_n} + \sum_{\forall \tau_j \in \Psi_m} \frac{c_j(1) + BW^{ub}}{p_j} > \theta$$

where the subset $\Psi_m$ contains the tasks on core $\mathcal{P}_m$ after assigning the first $(n-1)$ tasks. Note that $p^{min} \le p_i$ ($i = 1, \ldots, N$). Based on the definition of $\sigma$, the above inequality on every core $\mathcal{P}_m$ can be transformed as follows :

$$u_n(1) + 2\sigma + \sum_{\forall \tau_j \in \Psi_m} u_j(1) + |\Psi_m| \cdot \sigma > \theta$$

where $|\Psi_m|$ corresponds to the number of tasks in the subset $\Psi_m$. Adding up all these $M$ inequalities, there is:

$$(M-1) \cdot u_n(1) + \sum_{j=1}^{n} u_j(1) + (2M + n - 1) \cdot \sigma > M \cdot \theta.$$

By the assumption that tasks are sorted in non-ascending order of their level-1 utilizations, there is $u_n(1) \le \frac{\sum_{j=1}^{n} u_j(1)}{n}$. Then, the above inequality can be further transformed as:

$$\left(\frac{M-1}{n} + 1\right) \sum_{j=1}^{n} u_j(1) + (2M + n - 1) \cdot \sigma > M \cdot \theta$$

$$\Rightarrow \sum_{j=1}^{n} u_j(1) > \frac{n}{M + n - 1} \cdot [M \cdot \theta - M \cdot \sigma - (M + n - 1) \cdot \sigma].$$

Define $U(1)$ as the overall level-1 utilization of task set $\Psi$, that is, $U(1) = \sum_{j=1}^{N} u_j(1)$. As $U(1) \ge \sum_{j=1}^{n} u_j(1)$, we can further have:

$$U(1) > \frac{M \cdot n}{M + n - 1} \cdot (\theta - \sigma) - n \cdot \sigma = f(n) \tag{16}$$

where $f(n)$ is a function of $n$. Note that $\beta \cdot M + 1 \le n \le N$. To obtain a meaningful utilization bound, we need to have $f(n) > 0$, that is, $\frac{\theta-\sigma}{\sigma} > \frac{M+n-1}{M} \ge \frac{M+(\beta \cdot M+1)-1}{M} = 1 + \beta$. Therefore, we need to have $\sigma < \frac{\theta}{2+\beta} < 1$. Also, there is $\theta > \sigma$ because $\beta \ge 1$.

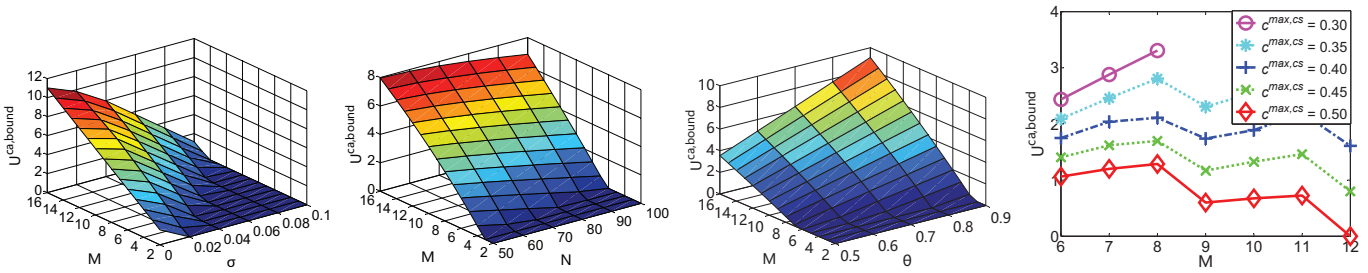
Since the values of $\theta$ and $\sigma$ are not related to $n$ and $M \ge 2$, we can get the second derivative of $f(n)$ with respect to $n$ as:

$$f''(n) = -\frac{2 \cdot M \cdot (M-1) \cdot (\theta - \sigma)}{(M + n - 1)^3} < 0.$$

Therefore, the function $f(n)$ is a concave function and thus its minimum value can be found when either $n = \beta \cdot M + 1$ or $n = N$:

$$f(\beta \cdot M + 1) = \frac{\beta \cdot M + 1}{1 + \beta} \cdot (\theta - \sigma) - (\beta \cdot M + 1) \cdot \sigma = U^{b1},$$

$$f(N) = \frac{M \cdot N}{M + N - 1} \cdot (\theta - \sigma) - N \cdot \sigma = U^{b2}.$$




(a) $\beta = 3$, $\theta = 0.9$ and $N = 100$ (b) $\beta = 3$, $\theta = 0.9$ and $\sigma = 0.05$ (c) $\beta = 4$, $\sigma = 0.03$ and $N = 100$ (d) Non-monotonicity of the bound

Fig. 1: Criticality-aware utilization bound and its non-monotonicity for partitioned-EDF and MSRP with the WFD mapping.

Hence, the level-1 utilization bound $U^{ca,bound}$ can be found as $U^{ca,bound} = \min\{U^{b1}, U^{b2}\}$, which concludes the proof. □

Note that, for non-MC task system when no task needs to access any resource, there are $\sigma = 0$ and $\theta = 1$. For the function $f(n)$ defined in Equation (16), we can get its first derivative with respect to $n$ as $f'(n) = \frac{M\cdot(M-1)}{(M+n-1)^2} > 0$. Therefore, the minimum value of $f(n)$ can be found as $\frac{\beta\cdot M+1}{\beta+1}$ when $n = \beta \cdot M + 1$, which exactly reduces to the bound for non-MC tasks scheduled under partitioned-EDF with WFD when no shared resource access is considered [38]. More specifically, for non-MC tasks that access shared resources under partitioned-EDF and MSRP with WFD (i.e., $\theta = 1$), the utilization bound represented by Equation (13) actually reduces to that as derived in [28].

### 5.2 Non-Monotonicity of the Utilization Bound

The relationship between the utilization bound $U^{ca,bound}$ and other system parameters can be more clearly shown in Figure 1.

From the figures, we can see that the utilization bound $U^{ca,bound}$ can decrease dramatically as the synchronization overhead factor $\sigma$ increases (e.g., when there are more and/or larger sized critical sections for tasks). For task systems with higher resource access contention (e.g., $\sigma > 0.05$), the bound can be extremely low as shown in Figure 1(a), which rather constrains its applicability. With other parameters being fixed, when there are more tasks (thus usually more synchronization requirements) in the system, $U^{ca,bound}$ can also decrease (see Figure 1(b)). In addition, when there are more available cores in the system and there are smaller $\omega$ and $K$ (i.e., a larger $\theta$ based on Property 9), $U^{ca,bound}$ can increase significantly as shown in Figure 1(c).

In general, when there are more available cores (i.e., a larger $M$), $U^{ca,bound}$ becomes larger. Nonetheless, for a given configuration of MC task set, the increased $\sigma$ (due to increased $M$) and thus increased synchronization overheads of tasks can in turn result in reduced $U^{ca,bound}$. Such *anomaly* can be more explicitly illustrated for task sets with $N = 50$, $p^{min} = 160$, $n^{max,cs} = 4$, $K = 2$, $\theta = 0.9$ and the maximum level-1 task utilization being 0.075 as shown in Figure 1(d). Here, for a given maximum length of critical sections among tasks (e.g., $c^{max,cs} = 0.45$), $U^{ca,bound}$ can become smaller when $M$ increases (e.g., from 8 to 9 and from 11 to 12). For the cases of $c^{max,cs} = 0.3$ ($c^{max,cs} = 0.35$ resp.), it turns out that there is $N \le \beta \cdot M$ when $M > 8$ ($M > 10$ resp.). Therefore, based on Lemma 2, any task set satisfying above configurations is guaranteed to be feasible under the partitioned-EDF scheduler and MSRP with the WFD mapping heuristic.

Below, we formally identify the *non-monotonicity* of the bound $U^{ca,bound}$. First, we can obtain lemmas related to $\beta$ as follows.

**Lemma 3.** *The minimum number of cores required to feasibly allocate task set $\Psi$ under partitioned-EDF is $M^{min} = \left\lceil \sum_{i=1}^{N} u_i(1) \right\rceil$.*

**Lemma 4.** *$\beta$ cannot decrease when the number of deployed cores decreases and other parameters are fixed.*

*Proof.* Note that $\beta = \lfloor \frac{\theta-\gamma}{\alpha} \rfloor$ and $\theta$ is not related to $M$. We can see that $\theta - \gamma = \theta - \frac{c^{max,cs}\cdot M\cdot K}{p^{min}}$ increases when $M$ decreases. For each task $\tau_i$, the function $\frac{c_i(1)+BW^{up}}{p_i} = \frac{c_i(1)+n^{max,cs}\cdot c^{max,cs}\cdot(M-1)}{p_i}$ decreases when $M$ decreases. Therefore, $\alpha$ also decreases when $M$ decreases. With its floor function being considered, $\beta$ cannot decrease when fewer cores are deployed, which concludes the proof. □

For a given $M$, due to the floor function of $\beta$, we can see that there exists an integer $\rho$ and an interval $[M, M-1, \cdots, M-\rho]$, in which $\beta$ remains invariant when $M$ decreases. We define such interval as the *constant interval* of $\beta$ and $\rho$ as its maximum *range*. A feasible method to compute $\rho$ is given by the lemma as follows.

**Lemma 5.** *For given number of cores $M$ ($\ge M^{min}$) and MC task set $\Psi$, the value of $\rho$ can be determined as:*

$$\rho = \min\{\max\{\lceil\varphi\rceil, 0\}, M - M^{min}\} \tag{17}$$

*where*

$$\varphi = \frac{(1+\beta)\cdot\alpha - (\theta-\gamma)}{\iota + (1+\beta)\cdot\lambda} - 1, \iota = \frac{K\cdot c^{max,cs}}{p^{min}},$$
$$\kappa + \lambda\cdot(M-1) = \max_{\forall\tau_i\in\Psi}\left\{\frac{c_i(1) + n^{max,cs}\cdot c^{max,cs}\cdot(M-1)}{p_i}\right\},$$

*and $\kappa$ and $\lambda$ are defined in the proof.*

*Proof.* Define $g(M)$ as $\alpha = \max_{\forall\tau_i}\{\frac{c_i(1)+n^{max,cs}\cdot c^{max,cs}\cdot(M-1)}{p_i}\}$ and $f(M)$ as $\frac{\theta-\gamma}{\alpha} = \frac{\theta-\iota\cdot M}{g(M)}$ where $\iota = \frac{K\cdot c^{max,cs}}{p^{min}}$. For given $M$ and MC task set $\Psi$, there must exist a task $\tau_i$ such that $\kappa + \lambda\cdot(M-1) = \alpha$, where $\kappa = \frac{c_i(1)}{p_i}$ and $\lambda = \frac{n^{max,cs}\cdot c^{max,cs}}{p_i}$. From the definition of $g(M)$, we can get $\kappa + \lambda\cdot(M-(\rho+1)-1) \le g(M-(\rho+1))$.

Note that, $\beta = \lfloor\frac{\theta-\gamma}{\alpha}\rfloor = \lfloor\frac{\theta-\iota\cdot M}{g(M)}\rfloor = \lfloor f(M)\rfloor$ and $\rho$ is defined as the range of constant interval for $\beta$. Based on Lemma 4, there is:

$$f(M-(\rho+1)) \ge 1+\beta \Rightarrow \frac{\theta-\iota\cdot(M-(\rho+1))}{g(M-(\rho+1))} \ge 1+\beta$$
$$\Rightarrow \frac{\theta-\gamma+\iota\cdot(\rho+1)}{\kappa+\lambda\cdot(M-(\rho+1)-1)} \ge 1+\beta \Rightarrow \frac{\theta-\gamma+\iota\cdot(\rho+1)}{\alpha-\lambda\cdot(\rho+1)} \ge 1+\beta$$
$$\Rightarrow \rho \ge \frac{(1+\beta)\cdot\alpha-(\theta-\gamma)}{\iota+(1+\beta)\cdot\lambda} - 1 = \varphi.$$

As $\rho \le M - M^{min}$ based on Lemma 3 and $\rho$ is a positive integer, we can get Equation (17) such that $f(M) < \ldots < f(M-\rho) \le \beta$ and $f(M-(\rho+1)) \ge \beta+1$, which concludes the proof. □

Based on the invariance of $\beta$ within its constant interval, we can obtain the sufficient condition for the non-monotonicity of the utilization bound $U^{ca,bound}$ as follows.

**Theorem 4.** *For a given set $\Psi$ of MC tasks that access shared resources in a multicore system with M cores, the criticality-aware utilization bound under partitioned-EDF and MSRP with WFD as given in Equation (13) can* decrease *when the number of deployed cores increases if there is $\frac{\theta}{3+\beta} \leq \sigma < \frac{\theta}{2+\beta}$.*

*Proof.* We define $\varsigma = \frac{n^{max,cs} \cdot c^{max,cs}}{p^{min}}$, $g(M) = U^{b1} = \frac{\beta \cdot M+1}{1+\beta} \cdot (\theta - \sigma) - (\beta \cdot M + 1) \cdot \sigma$ based on Equation (14) and $h(M) = U^{b2} = \frac{M \cdot N}{M+N-1} \cdot (\theta - \sigma) - N \cdot \sigma$ based on Equation (15).

First, suppose that $BW^{ub} \geq B^{ub}$. We can get $\sigma = \frac{BW^{ub}}{p^{min}} = \frac{(M-1) \cdot n^{max,cs} \cdot c^{max,cs}}{p^{min}} = (M-1) \cdot \varsigma$. Note that, $\varsigma$ and $\theta$ are not related to $M$, and $\beta$ can be considered as a constant based on Lemma 5. For $g(M)$, we can compute its first derivative with respect to $M$ as:

$$g'(M) = \beta \cdot \frac{\theta - \sigma}{1+\beta} - \varsigma \cdot \frac{\beta \cdot M + 1}{1+\beta} - (\beta \cdot M + 1) \cdot \varsigma - \beta \cdot \sigma.$$

Substitute $\varsigma = \frac{\sigma}{M-1}$ into the above equation, there is:

$$g'(M) = \beta \cdot \frac{\theta - \sigma}{1+\beta} - \beta \cdot \sigma - \frac{\beta \cdot M + 1}{M-1} \cdot \left( \frac{\sigma}{1+\beta} + \sigma \right).$$

As $\frac{\beta \cdot M+1}{M-1} > \beta$, the above equation can be transformed as:

$$g'(M) < \frac{\beta \cdot (\theta - 4\sigma - 2\beta \cdot \sigma)}{1+\beta}.$$

We can obtain $g'(M) < 0$ if $\frac{\theta}{2 \cdot (2+\beta)} \leq \sigma$. Moreover, from the proof in Theorem 3, we get $\sigma < \frac{\theta}{2+\beta}$. Therefore, when $\frac{\theta}{2 \cdot (2+\beta)} \leq \sigma < \frac{\theta}{2+\beta}$ holds, $g(M)$ decreases as $M$ increases.

For $h(M)$, its first derivative with regard to $M$ can be found as:

$$h'(M) = N \cdot \left( \frac{(N-1) \cdot (\theta - \sigma)}{(M+N-1)^2} - \frac{2M+N-1}{M+N-1} \cdot \varsigma \right).$$

Substitute $\varsigma = \frac{\sigma}{M-1}$ into the above equation, we can obtain:

$$\begin{aligned} h'(M) &= N \cdot \left( \frac{(N-1) \cdot (\theta - \sigma)}{(M+N-1)^2} - \frac{2M+N-1}{M+N-1} \cdot \frac{\sigma}{M-1} \right) \\ &< \frac{N}{M+N-1} \cdot \left( (\theta - \sigma) - 2 \cdot \sigma - \frac{(N-1) \cdot \sigma}{M-1} \right). \end{aligned}$$

As $N \geq \beta \cdot M + 1$, the above inequality can be simplified as:

$$h'(M) < \frac{N}{M+N-1} \cdot (\theta - 3 \cdot \sigma - \beta \cdot \sigma).$$

Therefore, we can get $h'(M) < 0$ when there is $\frac{\theta}{3+\beta} \leq \sigma$. Note that there is $\frac{1}{2 \cdot (2+\beta)} < \frac{1}{3+\beta}$ as $\beta \geq 1$. Hence, if there are $BW^{ub} \geq B^{ub}$ and $\frac{\theta}{3+\beta} \leq \sigma < \frac{\theta}{2+\beta}$, both $g(M)$ and $h(M)$ (thus the bound $U^{sc,bound}$) can decrease when the number of deployed cores $M$ increases.

For the other case where $BW^{ub} < B^{ub}$, by re-defining $\varsigma = \frac{c^{max,cs}}{p^{min}}$, there is $\sigma = \frac{B^{ub}}{p^{min}} = K \cdot M \cdot \varsigma$. For $g(M)$, its first derivative is:

$$\begin{aligned} g'(M) &= \beta \cdot \frac{\theta - \sigma}{1+\beta} - K \cdot \varsigma \cdot \frac{\beta \cdot M + 1}{1+\beta} - (\beta \cdot M + 1) \cdot K \cdot \varsigma - \beta \cdot \sigma \\ &\Rightarrow g'(M) = \beta \cdot \frac{\theta - \sigma}{1+\beta} - \beta \cdot \sigma - \frac{\beta \cdot M + 1}{M} \cdot \left( \frac{\sigma}{1+\beta} + \sigma \right). \end{aligned}$$

Similarly, for $h(M)$, its first derivative with respect to $M$ is:

$$h'(M) = N \cdot \left( \frac{(N-1) \cdot (\theta - \sigma)}{(M+N-1)^2} - \frac{2M+N-1}{M+N-1} \cdot \frac{\sigma}{M} \right).$$

Hence, following the similar steps, we can also get that both $g(M)$ and $h(M)$ decrease when $M$ increases if $\frac{\theta}{3+\beta} \leq \sigma < \frac{\theta}{2+\beta}$ holds. □

## 6 Bounds on the MC Tasks' Synchronization Overheads

From the above discussions, we can see that the rather pessimistic blocking items of tasks are incorporated in the utilization bound, e.g., the pessimistic $\omega$ is utilized for the level-1 core utilization limit as given in Equation (9), both $B^{ub}$ and $BW^{ub}$ are based on the largest size and the maximum number of critical sections for any task, etc. Such pessimistic synchronization overheads of tasks can significantly affect the bound and limit its usability. On the other hand, the non-monotonicity of the bound can cause anomalies in feasibility tests when more cores are deployed due to increased inter-core synchronization interference among tasks.

Recall that the *basic* computation of the maximum synchronization overheads of MC tasks is given in Section 4.2. To reduce the pessimism in the bound and improve tasks' schedulability, in what follows, we study a *criticality-aware* and *resource-oriented* analysis approach to tightening the bound on the synchronization overheads for MC tasks under partitioned EDF and MSRP.

Considering the fact that the system mode transitions are **stepwise** based on the AMC scheme, for any task, the basic steps of the new analysis approach are summarized as follows.

- We first get its maximum busy-waiting time at each level;
- Then, the bound of total global waiting time is reduced by considering its each resource (instead of critical section);
- Finally, the tightened bound on local blocking time can be obtained by reducing the scope of tasks that can block it.

### 6.1 Tightened Bound on the Global Waiting Time

Here, the first step is to consider a task's individual global waiting time **at each valid level**. For a given task partition, based on the AMC scheme and Property 1, the maximum global waiting time $BW_{i,x}(k)$ of task $\tau_i$ on core $\mathcal{P}_m$ for accessing resource $\mathcal{R}_a$ in its critical section $z_{i,x}$ ($r_{i,x} = \mathcal{R}_a$) at level-$k$ ($\leq \ell_i$) can be found as:

$$BW_{i,x}(k) = \sum_{j=1,\cdots,M}^{j \neq m} tp^{max}_{j,\mathcal{R}_a}(k) \tag{18}$$

where $tp^{max}_{j,\mathcal{R}_a}(k)$ denotes the maximum amount of time for any task on other core $\mathcal{P}_j$ ($j \neq m$) spent in accessing resource $\mathcal{R}_a$ once at level-$k$. Moreover, $tp^{max}_{j,\mathcal{R}_a}(k)$ can be further calculated as:

$$\begin{aligned} tp^{max}_{j,\mathcal{R}_a}(k) &= \max\{tt^{max}_{l,\mathcal{R}_a}(k) | \forall \tau_l \in \Psi_j : \ell_l \geq k\} \\ tt^{max}_{l,\mathcal{R}_a}(k) &= \max_{\forall \tau_l : \ell_l \geq k} \{c_{l,y} | \forall z_{l,y} : r_{l,y} = \mathcal{R}_a\} \end{aligned} \tag{19}$$

where $tt^{max}_{l,\mathcal{R}_a}(k)$ denotes the maximum amount of time for task $\tau_l$ ($\ell_l \geq k$) on a different core to access resource $\mathcal{R}_a$ once at level-$k$.

Based on $BW_{i,x}(k)$, as the largest size of any critical section is fixed, we can see that $BW_{i,x}(k) \geq BW_{i,x}(k+1)$ ($k = 1, \ldots, \ell_i - 1$). Note that, the system mode transition can happen during the execution of task $\tau_i$. Thus, by assuming that task $\tau_i$ is released at level-$k$ mode, $BW_{i,x}(k)$ essentially means the maximum amount of time that task $\tau_i$ waits before executing its critical section $z_{i,x}$ .

Hence, $BW_{i,x}$ can be found as $\max\{BW_{i,x}(k) | k = 1, \ldots, \ell_i\} = BW_{i,x}(1)$, which is the same as that in Equation (5). By exploiting such global waiting time at each valid level, we can tighten the bound on the local blocking times for MC tasks in next section.

Next, instead of simply accumulating $BW_{i,x}$ ($x = 1, \ldots, n_i$), we propose a *resource-oriented* method to reduce the worst-case **overall global waiting time** for task $\tau_i$. For such a purpose, we first analyze the limits on inter-core synchronization interference (due to accessing the same resource) for MC tasks on different

**Algorithm 1:** *CalBWMax*($\tau_i$ ($\in \Psi_m$), $k$ ($\leq \ell_i$), $\mathcal{R}_a$ ($\in \Phi_i$))

**Input:** $\tau_i$, $\mathbb{S}_a(k)$ and $\{\Psi_1, \cdots, \Psi_M\}$;
**Output:** $BW^{max}_{i,\mathcal{R}_a}(k)$;
1: $BW^{max}_{i,\mathcal{R}_a}(k) = 0$; $limit[x] = |\mathcal{S}_{i,a}|$ $(x = 1, \ldots, M \wedge x \neq m)$;
2: **for** $(z_{j,y} \in \mathbb{S}_a(k)$ $(\tau_j \in \Psi_x \wedge x \neq m))$ **do**
3: $num = \min\{\pi_{i,j}, limit[x]\}$;
4: **if** $(num == 0)$ **then**
5: continue;
6: **else**
7: $BW^{max}_{i,\mathcal{R}_a}(k) += num \cdot c_{j,y}$; $limit[x] -= num$;
8: **end if**
9: **end for**

cores. Recall that the initial arrival time of any task is 0. Then, the inter-core interference among tasks under partitioned EDF can be given by the following proposition [28].

**Proposition 3.** *For any two implicit-deadline tasks $\tau_i$ and $\tau_j$ on different cores scheduled under partitioned-EDF, the maximum number of $\tau_j$'s jobs that can interfere with the execution of any job of $\tau_i$ due to shared resource access is:*

$$\pi_{i,j} = \begin{cases} 1 & , \ p_i < p_j \wedge mod(p_j, p_i) = 0 \\ \frac{p_i}{p_j} & , \ p_i \geq p_j \wedge mod(p_i, p_j) = 0 \\ \left\lceil \frac{p_i}{p_j} \right\rceil + 1 & , \ otherwise \end{cases} \tag{20}$$

*where $mod(x, y)$ is defined as the remainder of dividing $x$ by $y$.*

Define $\Phi_i$ as the subset of resources that task $\tau_i$ needs to access. Note that, a MC task may access the same resource multiple times in its different critical sections. For a resource $\mathcal{R}_a$ ($\in \Phi_i$), we denote the subset of critical sections where task $\tau_i$ accesses it as $\mathcal{S}_{i,a} = \{z_{i,x} | \forall r_{i,x} = \mathcal{R}_a\}$. Then, based on Property 1, for any job of task $\tau_i$ on core $\mathcal{P}_m$, the number of interference from any other core $\mathcal{P}_x$ $(x \neq m)$ due to global waiting for accessing resource $\mathcal{R}_a$ at level-$k$ $(k \leq \ell_i)$ has the following two limits:

$|\mathcal{S}_{i,a}|$: the number of critical sections in the set $\mathcal{S}_{i,a}$, i.e., the number of times that any job of task $\tau_i$ accesses resource $\mathcal{R}_a$;

$\pi_{i,j}$: the maximum number of jobs of any task $\tau_j$ $(\ell_j \geq k)$ on $\mathcal{P}_x$ that need to access $\mathcal{R}_a$ during the execution of $\tau_i$'s any job.

With the above limits on inter-core interference among tasks being considered, we study a resource-oriented approach to reducing task $\tau_i$'s maximum total global waiting time $BW^{max}_{i,\mathcal{R}_a}(k)$ spent in accessing its **each resource** $\mathcal{R}_a$ at each valid level-$k$, rather than simply accumulating the maximum busy-waiting times of individual critical sections where task $\tau_i$ accesses resource $\mathcal{R}_a$.

We further define $\mathbb{S}_a(k)$ as the subset of critical sections of all tasks where $\mathcal{R}_a$ can be accessed at level-$k$, i.e., $\mathbb{S}_a(k) = \cup^{\ell_j \geq k}_{\forall \tau_j \in \Psi} \mathcal{S}_{j,a}$. Here, the critical sections in the subset $\mathbb{S}_a(k)$ are sorted in non-increasing order of their sizes and the tie is broken arbitrarily. Based on the definition of $\mathbb{S}_a(k)$, for any resource $\mathcal{R}_a$, we can obtain $\mathbb{S}_a(k) \supseteq \mathbb{S}_a(k+1)$ $(k = 1, \ldots, K-1)$.

For a given task-to-core mapping, the detailed steps to estimate the worst-case global waiting time $BW^{max}_{i,\mathcal{R}_a}(k)$ taken by task $\tau_i$ on $\mathcal{P}_m$ to access resource $\mathcal{R}_a$ at level-$k$ are summarized in Algorithm 1. Here, based on Proposition 1, the remaining limits on the number of interference from other cores are first initialized (line 1). Then, by non-ascending order of their sizes, the critical sections in $\mathbb{S}_a(k)$ are processed one at a time (line 2). For critical section $z_{j,y}$ where task $\tau_j$ accesses resource $\mathcal{R}_a$ on another core $\mathcal{P}_x$ $(x \neq m)$ (i.e., task $\tau_j$ can interfere with task $\tau_i$ when accessing resource $\mathcal{R}_a$ at level-$k$), its number of interference on $\tau_i$ is subject to the limits from $\tau_j$ (i.e., $\pi_{i,j}$) and the remaining limits from core $\mathcal{P}_x$ (i.e., $limit[x]$) (line 3). If there is no such interference (line 4), the next critical section in $\mathbb{S}_a(k)$ is to be processed (line 5); otherwise, $BW^{max}_{i,\mathcal{R}_a}(k)$ accumulates and the remaining interference limits from core $\mathcal{P}_x$ is accordingly updated (line 7). Finally, the resulting $BW^{max}_{i,\mathcal{R}_a}(k)$ is obtained and returned after all critical sections in the subset $\mathbb{S}_a(k)$ have been processed.

Based on the definition of $\mathbb{S}_a(k)$, for any resource $\mathcal{R}_a$, there is $BW^{max}_{i,\mathcal{R}_a}(k) \geq BW^{max}_{i,\mathcal{R}_a}(k+1)$ $(k = 1, \ldots, \ell_i - 1)$. Similarly, $BW^{max}_{i,\mathcal{R}_a}(k)$ indicates the maximum total busy-waiting time for task $\tau_i$ to access resource $\mathcal{R}_a$, providing that task $\tau_i$ arrives at level-$k$ mode.

Therefore, the worst-case global waiting time $BW_i(k)$ for task $\tau_i$ to access all its resources required at level-$k$ can be found as:

$$BW_i(k) = \sum_{\forall \mathcal{R}_a \in \Phi_i} BW^{max}_{i,\mathcal{R}_a}(k). \tag{21}$$

Finally, the bound on the overall global waiting time of task $\tau_i$ spent in accessing all its resources can be tightened as:

$$BW_i = \max\{BW_i(k) | k = 1, \ldots, \ell_i\} = BW_i(1). \tag{22}$$

Recall that the maximum number of critical sections for a task is $n^{max,cs}$ as defined in Section 5.1. We can see that for any resource $\mathcal{R}_a$ that task $\tau_i$ needs to access, there are at most $N \cdot n^{max,cs}$ critical sections in both the set $\mathbb{S}_a(k)$ and the task system considered. Thus, the time complexity of both Algorithm 1 and the computation of $BW_i$ for task $\tau_i$ (see Equation (22)) can be found as $O(N \cdot n^{max,cs})$.

### 6.2 Tightened Bound on the Local Blocking Time

Based on the analysis in Section 4.1, task $\tau_i$ can be blocked due to pi-blocking at most once and such blocking can occur at any criticality level-$k$ ($\leq \ell_i$) *regarding to pi-blocking*. After that, task $\tau_i$ cannot experience pi-blocking any more but can *only* encounter ci-blocking at most once during mode-change period at its each *valid* criticality level-$k$ ($k < \ell_i$) *regarding to ci-blocking*.

Hence, to avoid pessimistic estimates of synchronization overheads due to repeated computation of blocking items, the key point to reduce the bound on the local blocking time for a MC task is: more precisely ascertaining **the scope of tasks** that can block it at each valid level regarding to pi-blocking and ci-blocking.

With the focus on the step-wise system mode transitions, we first consider the scenarios where pi-blocking for a task occurs **at its each valid level** individually. Assume that task $\tau_i$ executes on core $\mathcal{P}_m$ and the system operates at level-$k$ ($\leq \ell_l$). Based on the reduced global waiting times of tasks discussed in Section 6.1, for a given task-to-core partition, the maximum local pi-blocking time for task $\tau_i$ on core $\mathcal{P}_m$ in normal mode at level-$k$ can be found as:

$$\begin{aligned} B^{pi}_i(k) = \max\{&BW_{j,y}(k) + c_{j,y} | \forall z_{j,y} : \tau_j \in \Psi_m \\ &\wedge \tau_j \in \{lpH(\tau_i) \cup lpL(\tau_i)\} \wedge \ell_j \geq k\} \end{aligned} \tag{23}$$

where $BW_{j,y}(k)$ is defined in Equation (18). Recall that if $\tau_j \in \{lpH(\tau_i) \cup lpL(\tau_i)\}$, there is $p_j > p_i$. When there is no such task $\tau_j$ on core $\mathcal{P}_m$ at level-$k$, we can have $B^{pi}_i(k) = 0$.

We can obtain the properties with respect to the ci-blocking for MC tasks under partitioned-EDF and MSRP as follows.

**Property 10.** *If $\tau_i$ encounters pi-blocking by a task ($\in lpH(\tau_i)$) in normal mode at level-k, it cannot experience ci-blocking by any task ($\in lpL(\tau_i)$) under partitioned-EDF and MSRP.*

TABLE 1: The timing parameters of MC tasks and the maximum synchronization overheads based on the basic analysis approach

| $\tau_i$ | $\ell_i$ | $c_i(\ell_i)$ | $p_i$ | $n_i$ | $\{r_{i,x}\}(x=1,\ldots,n_i)$ | $\{c_{i,x}\}(x=1,\ldots,n_i)$ | $\frac{c_i(\ell_i)}{p_i}$ | $P(\tau_i)$ | $BW_{i,1}$ | $BW_{i,2}$ | $BW_i$ | $B_i^{pi}$ | $B_i^{ci}(1)$ | $B_i^{ci}(2)$ | $B_i^{ci}$ | $B_i$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\tau_1$ | 1 | 13 | 71 | 1 | $\{\mathcal{R}_1\}$ | {6} | 0.183 | $\mathcal{P}_1$ | 5 | – | 5 | 0 | – | – | 0 | 0 |
| $\tau_2$ | 3 | 11 | 57 | 1 | $\{\mathcal{R}_2\}$ | {2} | 0.193 | $\mathcal{P}_1$ | 0 | – | 0 | 11 | 11 | 0 | 11 | 22 |
| $\tau_3$ | 3 | 19 | 62 | 2 | $\{\mathcal{R}_1,\mathcal{R}_3\}$ | {3, 1} | 0.307 | $\mathcal{P}_1$ | 5 | 5 | 10 | 11 | 11 | 0 | 11 | 22 |
| $\tau_4$ | 2 | 22 | 72 | 2 | $\{\mathcal{R}_3,\mathcal{R}_1\}$ | {5, 2} | 0.306 | $\mathcal{P}_2$ | 1 | 6 | 7 | 11 | 11 | – | 11 | 22 |
| $\tau_5$ | 1 | 13 | 88 | 2 | $\{\mathcal{R}_1,\mathcal{R}_1\}$ | {5, 1} | 0.148 | $\mathcal{P}_2$ | 6 | 6 | 12 | 0 | – | – | 0 | 0 |
| $\tau_6$ | 1 | 15 | 62 | 2 | $\{\mathcal{R}_3,\mathcal{R}_3\}$ | {5, 1} | 0.242 | $\mathcal{P}_2$ | 1 | 1 | 2 | 11 | – | – | 0 | 11 |

*Proof.* Assume that task $\tau_i$ experiences pi-blocking by task $\tau_x$ ($\in lpH(\tau_i)$) in normal mode at level-$k$, and experiences ci-blocking by another task $\tau_y$ ($\in lpL(\tau_i)$) during mode-change period at level-$l$ ($l < \ell_i$). From the definition of the subset $lpL(\tau_i)$, task $\tau_i$ is blocked due to pi-blocking twice, which contradicts Property 4. □

**Property 11.** *If $\tau_i$ encounters pi-blocking by a task ($\in lpL(\tau_i)$) in normal mode at level-k, it cannot experience ci-blocking by another task ($\in lpL(\tau_i)$) under partitioned-EDF and MSRP.*

**Property 12.** *Suppose that $\tau_i$ is blocked by a task $\tau_j$ ($\in lpL(\tau_i)$) during mode-change period at level-k. Task $\tau_i$ can be blocked* only *by task $\tau_j$ due to both pi-blocking and ci-blocking at level-k under the partitioned-EDF scheduler with MSRP.*

Properties 10 to 12 indicate that once $\tau_i$ encounters pi-blocking at level-$k$, it can experience ci-blocking *only* by tasks pertaining to the subset $hpL(\tau_i)$ on core $\mathcal{P}_m$. Therefore, we can obtain the theorem with respect to the maximum ci-blocking time for task $\tau_i$ at its each valid level regarding to ci-blocking as follows.

**Theorem 5.** *Suppose that task $\tau_i$ experiences pi-blocking on core $\mathcal{P}_m$ at level-k. The maximum ci-blocking time $B_i^{ci}(x)$ for task $\tau_i$ at any valid level-x regarding to ci-blocking can be found as:*

$$B_i^{ci}(x) = \max\{BW_{j,y}(x) + c_{j,y} | \forall z_{j,y} : \tau_j \in \Psi_m \wedge \tau_j \in hpL(\tau_i) \wedge \ell_j = x\}, x = 1, \ldots, \ell_i - 1 \quad (24)$$

where $B_i^{ci}(x) = 0$ if there is no such task $\tau_j$. By exploiting the busy-waiting time of any task at each valid level in Equation (18) and considering the reduced range of tasks that can cause ci-blocking for task $\tau_i$, we can see that Equation (24) typically represents a less pessimistic estimate of ci-blocking time compared to Equation (8).

Hence, by assuming that pi-blocking for task $\tau_i$ occurs at level-$k$, the maximum local blocking time $B_i(k)$ for $\tau_i$ can be found as $B_i(k) = B_i^{pi}(k) + \sum_{x=1}^{\ell_i-1} B_i^{ci}(x)$. As pi-blocking for $\tau_i$ can occur at any level-$k$ ($\leq \ell_i$), the maximum local blocking time for $\tau_i$ is computed as $B_i = \max\{B_i(k) | k = 1, \ldots, \ell_i\}$ and can be further simplified as:

$$B_i = B_i^{pi} + B_i^{ci} = max\{B_i^{pi}(k) | k = 1, \ldots, \ell_i\} + \sum_{x=1}^{\ell_i-1} B_i^{ci}(x). \quad (25)$$

Next, for the other case where **no pi-blocking** occurs during the execution of $\tau_i$, only ci-blocking for $\tau_i$ needs to be considered. Here, the maximum local blocking time for task $\tau_i$ is denoted as $\mathfrak{B}_i$. Since there is no pi-blocking for task $\tau_i$, only tasks in the subset $hpL(\tau_i)$ on core $\mathcal{P}_m$ can cause ci-blocking for $\tau_i$. Therefore, we can have $\mathfrak{B}_i = \sum_{x=1}^{\ell_i-1} B_i^{ci}(x)$ where $B_i^{ci}(x)$ is defined in Equation (24), and we can directly obtain the following property.

**Property 13.** *For a given task-to-core mapping, there is $B_i \geq \mathfrak{B}_i$ for any MC task $\tau_i$ scheduled under partitioned-EDF and MSRP.*

Finally, taking the worst-case scenarios for the synchronization overheads into consideration, the *new* tightened bound for the local blocking time experienced by any MC task under partitioned EDF and MSRP can be obtained based on Equations (18) and (23)-(25).

**An Example:** We consider six MC tasks with three criticality levels that run on a dual-core system with three shared resources. The resource access patterns of tasks are shown in Figure 2 and their timing parameters are provided in Table 1, where $P(\tau_i)$ refers to the core where task $\tau_i$ is allocated.

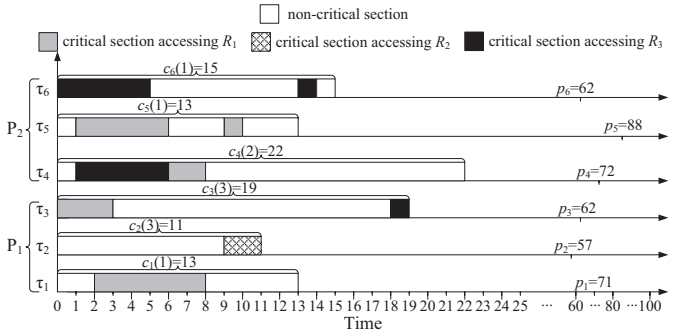



Fig. 2: An example of MC tasks and resource access patterns.

The task-to-core partition follows the WFD heuristic: tasks are ordered by their descending maximum utilizations at their own criticality and the tie is broken by favoring the task with smaller index, which is $\tau_3, \tau_4, \tau_6, \tau_2, \tau_1, \tau_5$; the tasks are processed one at a time; any task is assigned to the core with the maximum available utilization and thus there are $\Psi_1 = \{\tau_1, \tau_2, \tau_3\}$ and $\Psi_2 = \{\tau_4, \tau_5, \tau_6\}$.

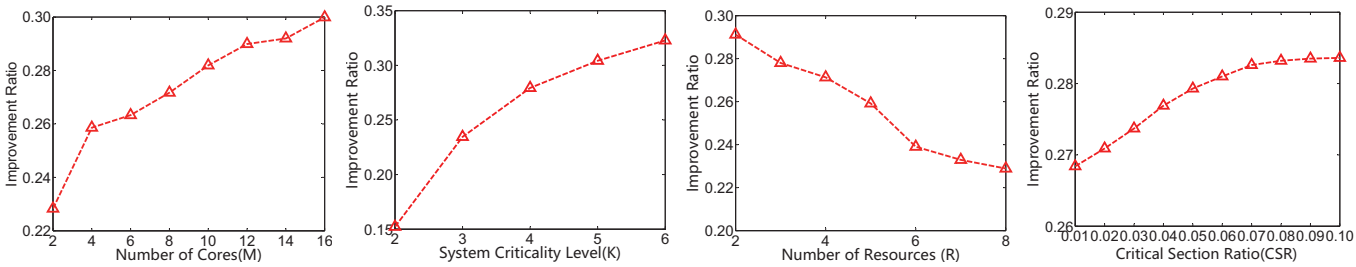
First, we consider the *basic analysis approach* to computing the synchronization overheads of tasks as presented in Section 4.2. For task $\tau_2$ on core $\mathcal{P}_1$, we can get $BW_{2,1} = 0$ from Equation (5), $BW_2 = 0$, $B_2^{pi} = \max\{BW_{1,1} + c_{1,1}, BW_{3,1} + c_{3,1}, BW_{3,2} + c_{3,2}\} = \max\{5+6, 5+3, 5+1\} = 11$ from Equation (7), and $B_2^{ci} = B_2^{ci}(1) = \max\{BW_{1,1} + c_{1,1}\} = 5 + 6 = 11$ from Equation (8). Hence, the maximum blocking time for $\tau_2$ is $B_2 = B_2^{pi} + B_2^{ci} = 22$. Following the similar steps, the maximum synchronization overheads of other tasks can be obtained and are shown in Table 1. For task $\tau_3$ on core $\mathcal{P}_1$, we can have $\frac{B_3}{p_3} + \frac{c_2(3)+BW_2}{p_2} + \frac{c_3(3)+BW_3}{p_3} = 1.016 > 1$, which violates the schedulability condition in Equation (4).

Next, we consider the *new analysis approach* as discussed in this section. Based on Equation (18), $BW_{i,x}(k)$ ($i = 1, \ldots, 6; x = 1, 2; k = 1, 2, 3$) for any critical section of any task at its any valid level can be obtained and are shown in Table 2. Note that, based on Algorithm 1 and Equations (20) to (22), we can have $BW_6 = 1$, because task $\tau_3$ on core $\mathcal{P}_1$ has the same period as $\tau_6$ and thus $\tau_3$ interferes with $\tau_6$ only once when $\tau_6$ accesses resource $\mathcal{R}_3$ twice.

For task $\tau_4$ on core $\mathcal{P}_2$, based on Equation (23), there is $B_4^{pi} = B_4^{pi}(1) = \max\{BW_{5,1}(1) + c_{5,1}, BW_{5,2}(1) + c_{5,2}\} = \max\{6+5, 6+1\} = 11$. Also, based on Equation (24), we can have $B_4^{ci}(1) = \max\{BW_{6,1}(1)+c_{6,1}, BW_{6,2}(1)+c_{6,2}\} = \max\{1+5, 1+1\} = 6$ because there is only one task (i.e., $\tau_6$) in the subset $hpL(\tau_4)$ on core $\mathcal{P}_2$

TABLE 2: The tightened bound of synchronization overheads based on the new analysis approach

| $\tau_i$ | $BW_{i,1}(1)$ | $BW_{i,1}(2)$ | $BW_{i,1}(3)$ | $BW_{i,2}(1)$ | $BW_{i,2}(2)$ | $BW_{i,2}(3)$ | $BW_i$ | $B_i^{pi}(1)$ | $B_i^{pi}(2)$ | $B_i^{pi}(3)$ | $B_i^{pi}$ | $B_i^{ci}(1)$ | $B_i^{ci}(2)$ | $B_i^{ci}$ | $B_i$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\tau_1$ | 5 | – | – | – | – | – | 5 | 0 | – | – | 0 | – | – | 0 | 0 |
| $\tau_2$ | 0 | 0 | 0 | – | – | – | 0 | 11 | **6** | **3** | 11 | **0** | 0 | **0** | **11** |
| $\tau_3$ | 5 | 2 | 0 | 5 | 5 | 0 | 10 | 11 | 0 | 0 | 11 | **0** | 0 | **0** | **11** |
| $\tau_4$ | 1 | 1 | – | 6 | 3 | – | 7 | 11 | 0 | – | 11 | **6** | – | **6** | **17** |
| $\tau_5$ | 6 | – | – | 6 | – | – | 12 | 0 | – | – | 0 | – | – | 0 | 0 |
| $\tau_6$ | 1 | – | – | 1 | – | – | **1** | 11 | – | – | 11 | – | – | 0 | 11 |




(a) Improvement with varying $M$ (b) Improvement with varying $K$ (c) Improvement with varying $R$ (d) Improvement with varying $CSR$

Fig. 3: Performance of the new analysis approach to reducing the bound on local blocking time.

when system operates at level-1. As the own criticality level of task $\tau_4$ is 2, we can get $B_4^{ci} = B_4^{ci}(1) = 6$ and $B_4 = B_4^{pi} + B_4^{ci} = 11+6 = 17$ based on Equation (25). Following these steps, the synchronization overheads of other tasks can be calculated as shown in Table 2. Based on the schedulability condition in Equation (4), all tasks are feasible with the reduced bound of synchronization overheads.

## 7 Performance Evaluations and Discussions

To the best of our knowledge, the scheduling of MC tasks that access shared resources in multicores under EDF-VD [24], [35] is *unexplored* so far. Here, we evaluate the performance of the new analysis approach and partitioned-EDF with MSRP: the simulations in Section 7.2 assess the new analysis approach in terms of blocking time reduction and schedulability ratio improvement; while the implementation in Linux (see Section 7.3) measures the usability of partitioned-EDF with MSRP regarding to the run-time overhead. Moreover, in the experiments, the task-to-core mapping follows *WFD* without considering shared resource access.

### 7.1 Parameter Settings

The full ranges for system parameters are first provided in Table 3, including the number of cores $M$, the system criticality level $K$, the normalized system utilization $NSU$ (defined as the ratio of total maximum utilization of tasks at their own criticality to the number of cores) and the number of shared resources $R$. Table 3 also shows full ranges for task parameters, i.e., the number of tasks $N$, the periods of tasks $P$, the critical section ratio $CSR$ (defined as the overall size of critical sections in a task over its maximum WCET) and the number of critical sections $n_i$ in any task $\tau_i$.

The synthetic task sets are generated as follows. The number of shared resources in the system $R$ is uniformly selected in the range $[2, 8]$. For given $NSU$, $M$ and $N$, the base maximum utilization of $\tau_i$ is set as $u_i^{base} = \frac{NSU \cdot M}{N}$. Then, its period is randomly selected in one of the three period ranges as given in Table 3, its own criticality level $\ell_i$ is chosen uniformly within $[1, K]$ and $c_i(\ell_i)$ is obtained uniformly in the range $[0.2 \cdot p_i \cdot u_i^{base}, 1.8 \cdot p_i \cdot u_i^{base}]$. Next, the number of critical sections $n_i$ in task $\tau_i$ is obtained uniformly in the range $[1, 16]$ where the resource accessed in each critical section is randomly selected. Finally, the size of a critical section is generated uniformly within $\left[\frac{0.2 \cdot c_i(\ell_i) \cdot CSR}{n_i}, \frac{1.8 \cdot c_i(\ell_i) \cdot CSR}{n_i}\right]$.

TABLE 3: System and task parameters for the experiments

| Parameters | Values/ranges |
|---|---|
| Number of cores ($M$) | $[2, 16]$ |
| System criticality level ($K$) | $[2, 6]$ |
| Normalized system utilization ($NSU$) | $[0.3, 0.78]$ |
| Number of resources ($R$) | $[2, 8]$ |
| Number of tasks ($N$) | $[20, 160]$ |
| Task periods ($P$) | [50, 200],[200, 500],[500, 2000] |
| Critical section ratio ($CSR$) | $[0.01, 0.1]$ |
| Number of critical sections in a task ($n_i$) | $[1, 16]$ |

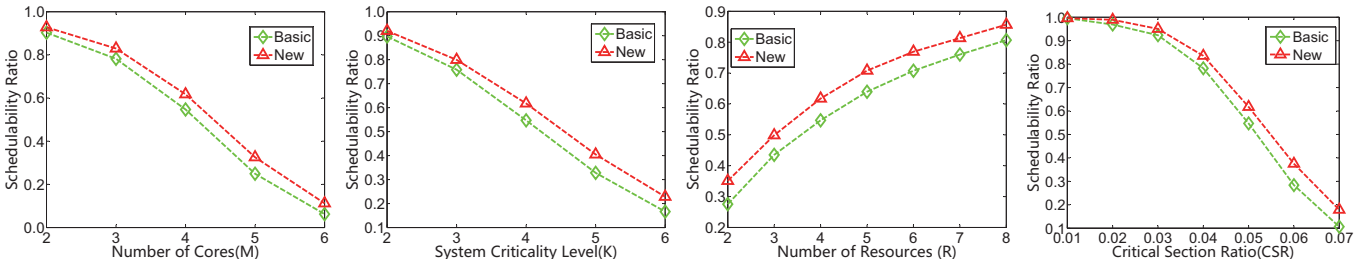
### 7.2 Simulation Performance of the New Approach

Unless otherwise noted, the default values of parameters are: $M = 4$, $N = 40$, $K = 4$, $NSU = 0.72$, $R = 4$ and $CSR = 0.05$. For the results reported below, each data point corresponds to the average result of 30,000 task sets. Moreover, the number of tasks $N$ is set as $10 \cdot M$ when $M$ can vary as shown in Figures 3(a) and 4(a) .

#### 7.2.1 Reduction in Local Blocking Time

Figure 3 shows the performance of the *new* analysis approach in terms of the percent of reduced blocking time (i.e., improvement ratio) compared to the *basic* analysis approach (see Section 4.2).

In general, more available cores (i.e., larger $M$) with fixed number of tasks per core, higher system criticality level (i.e., larger $K$), fewer shared resources (i.e., smaller $R$) and larger size of critical sections (i.e., larger $CSR$) represent higher resource access contention among tasks and more synchronization requirements in the system. Recall that for any MC task, the new approach computes its individual global waiting time at each level and reduces the scope of tasks that can block it at each valid level regarding to ci-blocking. Therefore, for such cases, the new approach can effectively reduce the bound on the local blocking times for tasks, and thus can achieve more improvements (up to 30%) compared to the basic approach as shown in Figures 3(a) to 3(d).




(a) Schedulability with varying $M$ (b) Schedulability with varying $K$ (c) Schedulability with varying $R$ (d) Schedulability with varying $CSR$

Fig. 4: Schedulability performance of the two analysis approaches.

#### 7.2.2 Schedulability Performance

Figure 4 further shows the schedulability comparison for the two approaches based on the schedulability condition in Equation (4).

Similar to the trends for blocking time reduction as shown in Figure 3, when other parameters are fixed, larger values of $M$, $K$ and $CSR$ and smaller value of $R$ normally correspond to more synchronization requirements and thus degraded acceptance ratios for the two approaches. Recall that, the new analysis approach tries to reduce the maximum total global waiting times of tasks using a resource-oriented method as presented in Section 6.1. Moreover, as explained above, the new approach can tighten the bound on the local ci-blocking times for tasks by exploiting the busy-waiting time of any task at its each valid level. Note surprisingly, for such cases, the new analysis approach can achieve better schedulability ratio (e.g., 10% more) as shown in Figures 4(a) to 4(d).

### 7.3 Empirical Results for partitioned-EDF with MSRP

To measure the usability of the partitioned-EDF scheduler with MSRP empirically, we implemented them in Linux kernel 2.6.38.8, which run on a PC with a 32nm AMD FX-8320 processor (8 cores, 3.5Ghz clock speed, 8M L2 and L3 cache) and 8G RAM.

We followed the design paradigms of LITMUS$^{RT}$ [18] and exploited the available Linux infrastructure to establish partitioned EDF scheduler: a new scheduling class with the highest priority is added to the traditional Linux scheduler. We also provided a user space library to create MC tasks by means of multi-threading. The tasks are initially created as non real-time where each task executes the same function codes by accumulating a local variable in while-loop to stress the OS. The system call is used to pass the timing parameters of tasks from user space to kernel space and then per-task data structures are constructed in kernel.

Specifically, we implemented two *additional global synchronization mechanisms* designed for MC systems: the first is used to synchronize the tick counters for cores; based on the first one, the second mechanism is exploited to address the issues when the system mode transition occurs, such as dropping low-criticality jobs upon exiting all their criticality sections (if possible).

**Parameter Settings:** The period range for any task is set to [50$ms$, 1$s$] and each task set executes for 10 seconds. Moreover, there are two additional parameters necessary for the measurements.

The first is *increment factor* ($IFC$) defined as the increasing ratio of WCETs between consecutive criticality levels for any task. Following the similar steps to generate task sets in Section 7.1, for any task $\tau_i$, we first obtain its $c_i(\ell_i)$ and then its WCETs at other valid levels can be generated according to $c_i(\ell_i)$ and $IFC$.

The second is *mode-change probability* ($MCP$) that is ranged from 0.01 to 0.1 and accounts for the online execution variations of tasks. We first compute the total number of jobs for all tasks executed in 10 seconds, which is multiplied by $MCP$ for obtaining the number of jobs that can cause mode transition. After randomly selecting such jobs (with their own criticality levels being higher than 1), the actual execution times of these jobs can be uniformly chosen from their minimum WCETs to the maximum ones.

The metric of measurement performance is run-time overhead, including context switching overhead, scheduling overhead caused by the operations for individual job queues of cores and resources, the additional overhead for global synchronization mechanisms, etc. The default parameter values are $IFC = 0.2$, $MCP = 0.05$ and the others are the same as those in the simulations. For the results given below, each data point represents the average result of 1000 *schedulable* task sets based on the new analysis approach and the schedulability condition given in Equation (4).

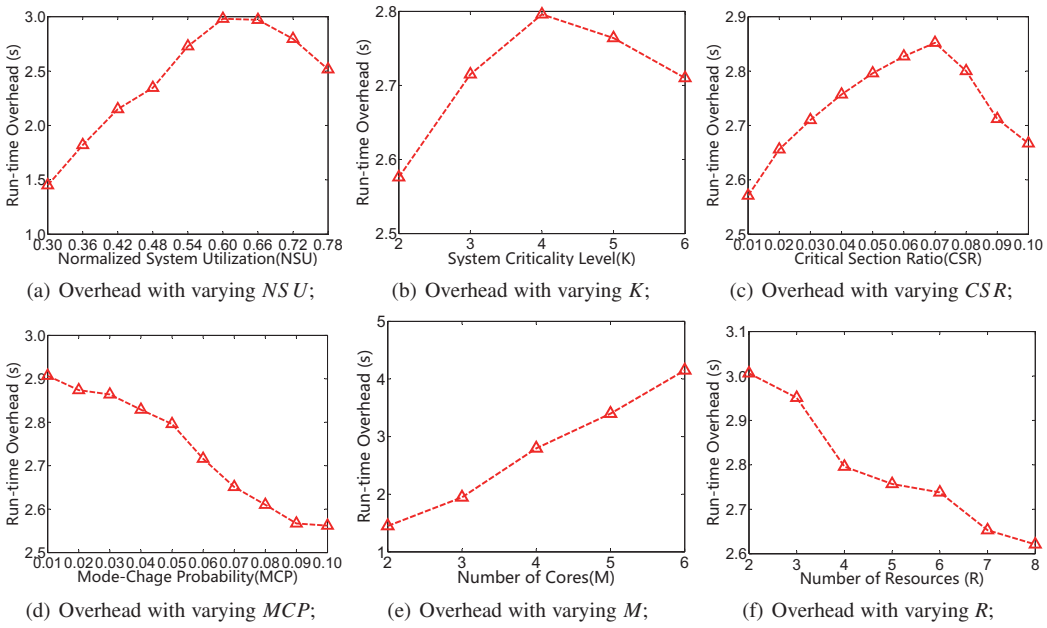
**Empirical Results:** The measurement results for the run-time overhead (in $s$) of partitioned-EDF with MSRP are shown in Figure 5. The results show its applicability: the overhead normally consumes about 3% to 7% of the total execution time, which is approximate to that (i.e., 5%) for partitioned EDF-VD [24] on Core i5 platform where no shared resource is considered [7].

Figure 5(a) shows the measured overhead for partitioned-EDF and MSRP with varying normalized system utilizations ($NSU$). In general, a larger $NSU$ means heavier workload and thus higher run-time overhead. Therefore, when $NSU$ increases, higher overhead can be experienced by partitioned-EDF with MSRP. Note that, the mode-change probability $MCP$ is relatively large here (i.e., 0.05). This indicates that when system workload becomes heavier, the tasks with higher utilizations can be discarded earlier due to the AMC scheme. For this reason, when $NSU$ reaches a large value (e.g., 0.66), the overhead can decrease.

Next, the impacts of different system criticality levels ($K$) on the overhead for partitioned-EDF with MSRP are evaluated and the results are shown in Figure 5(b). Generally, a larger $K$ implies higher utilizations of tasks at higher levels and thus higher overhead. Similar to the trends as those for varying $NSU$, when $K$ increases, the partitioned-EDF scheduler with MSRP has higher overhead; whereas when $K$ reaches a large value (e.g., 5), the overhead incurred by partitioned-EDF with MSRP can decrease.

Figure 5(c) shows the overhead for partitioned-EDF with MSRP when the critical section ratio ($CSR$) can vary. Generally, larger size of critical sections in tasks (i.e., a larger $CSR$) can result in more synchronization requirements and thus higher overhead caused by the operations for job queues of resources. With a relative large $MCP$ being considered, the overhead for partitioned-EDF with MSRP usually increases when $CSR$ increases and can decrease when $CSR$ reaches a large value (e.g., 0.08).

We further evaluate the impacts of mode-change probability




(a) Overhead with varying $NSU$; (b) Overhead with varying $K$; (c) Overhead with varying $CSR$;

(d) Overhead with varying $MCP$; (e) Overhead with varying $M$; (f) Overhead with varying $R$;

Fig. 5: Run-time overhead of partitioned-EDF with MSRP for MC tasks running on multicores with shared resources.

($MCP$) on the measured overhead for partitioned-EDF with MSRP and the results are shown in Figure 5(d). With other parameters being fixed, when $MCP$ increases, more tasks could be dropped that leads to less context switching, fewer operations for job queues of cores and resources (and thus lower overhead). Nonetheless, when $MCP$ becomes large (e.g., $MCP > 0.8$), the overhead reduction begins to stagnate due to the AMC scheme: some jobs of the same own criticality that are expected to cause running mode transition multiple times can be dropped simultaneously.

Figure 5(e) shows the overhead performance for partitioned-EDF with MSRP when the number of cores ($M$) can vary. Here, the number of tasks $N$ is set as $5 \cdot M$ to measure the overheads caused by the global synchronization mechanisms designed for MC systems as well as the operations for job queues of resources. Not surprisingly, the overhead incurred by partitioned-EDF with MSRP increases when there are more available cores in the system due to more operations for job queues of resources and more times consumed to address the issues for system mode transitions.

Finally, the overhead performance for partitioned-EDF with MSRP with different number of shared resources ($R$) is shown in Figure 5(f). When other parameters are fixed, as more available shared resources commonly account for relatively less resource access contention among tasks, the overhead experienced by partitioned EDF with MSRP typically decreases when $R$ increases.

## 8 Conclusion

For periodic mixed-criticality tasks that access shared resources in multicores, we characterize the blocking issues due to both priority and criticality inversions for MSRP under the partitioned EDF scheduler. Based on level-1 utilization limit on any core, we study a criticality-cognizant utilization bound under partitioned EDF and MSRP with WFD. Then, we discuss its non-monotonicity with regard to the number of cores, which implies that the schedulability performance may be improved when fewer cores are deployed due to the reduced inter-core synchronization interference.

With the focus on the step-wise system mode transitions, we develop a novel analysis approach to computing the individual global waiting time of a task at its each valid level, and then exploit a resource-oriented technique to tighten the bound of synchronization overheads for MC tasks scheduled by partitioned-EDF with MSRP. The extensive simulation results and empirical results validate the effectiveness (i.e., better schedulability ratio) of the new analysis approach and the usability (i.e., acceptable run-time overhead) of the partitioned-EDF scheduler with MSRP.

## Acknowledgments

This work is supported in part by the National Natural Science Foundation of China (NSFC) Awards 61472150 and 61173045, the Fundamental Research Funds for the Central Universities (China) Awards HUST: 2016YXMS081 and 2015TS072, US National Science Foundation Awards CNS-1422709 and CNS-1421855.

## References

[1] I. RTCA, "Do-178c: Software considerations in airborne systems and equipment certification," 2011.

[2] S. Vestal, "Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance," in *Proc. of the IEEE Real-Time Systems Symposium*, pp. 239–243, 2007.

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

[4] Z. Guo, L. Santinelli, and K. Yang, "Edf schedulability analysis on mixed-criticality systems with permitted failure probability," in *Proc. of IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, pp. 187–196, 2015.

[5] L. S. Indrusiak, J. Harbin, and A. Burns, "Average and worst-case latency improvements in mixed-criticality wormhole networks-on-chip," in *Proc. of the Euromicro Conference on Real-Time Systems*, pp. 47–56, 2015.

[6] A. Kostrzewa, S. Saidi, and R. Ernst, "Dynamic control for mixed-critical networks-on-chip," in *Proc. of Real-Time Systems Symposium*, pp. 317–326, 2015.

[7] L. Sigrist, G. Giannopoulou, P. Huang, A. Gomez, and L. Thiele, "Mixed-criticality runtime mechanisms and evaluation on multicores," in *Proc. of Real-Time and Embedded Technology and Applications Symposium*, pp. 194–206, 2015.

[8] N. Zhang, C. Xu, J. Li, and P. M, "A sufficient response-time analysis for mixed criticality systems with pessimistic period," *Journal of Computational Information Systems*, vol. 11, no. 6, pp. 1955–1964, 2015.

[9] T. P. Baker, "Stack-based scheduling for realtime processes," *Real-Time System*, vol. 3, no. 1, pp. 67–99, 1991.

[10] L. Sha, R. Rajkumar, and J. P. Lehoczky, "Priority inheritance protocols: An approach to real-time synchronization," *IEEE Transactions on Computer*, vol. 39, no. 9, pp. 1175–1185, 1990.

[11] A. Block, H. Leontyev, B. B. Brandenburg, and J.-H. Anderson, "A flexible real-time locking protocol for multiprocessors," in *Proc. of the IEEE Int'l Conference on Embedded and Real-Time Computing Systems and Applications*, 2007.

[12] P. Gai, M. Di Natale, G. Lipari, A. Ferrari, C. Gabellini, and P. Marceca, "A comparison of mpcp and msrp when sharing resources in the janus multiple-processor on a chip platform," in *Proc. of Real-Time and Embedded Technology and Applications Symposium*, pp. 189–198, 2003.

[13] K. Lakshmanan, D.-d. Niz, and R. Rajkumar, "Coordinated task scheduling, allocation and synchronization on multiprocessors," in *Proc. of the IEEE Real-Time Systems Symposium*, pp. 469–478, 2009.

[14] R. Rajkumar, "Real-time synchronization protocols for shared memory multiprocessors," in *Porc. of the International Conference onDistributed Computing Systems*, pp. 116–123, 1990.

[15] A. Wieder and B. B. Brandenburg, "On spin locks in autosar: Blocking analysis of fifo, unordered, and priority-ordered spin locks," in *Proc. of Real-Time Systems Symposium*, pp. 45–56, 2013.

[16] S. Bunzel, "Autosar–the standardized software architecture," *Informatik-Spektrum*, vol. 34, no. 1, pp. 79–83, 2011.

[17] M. Yang, A. Wieder, and B. B. Brandenburg, "Global real-time semaphore protocols: A survey, unified analysis, and comparison," in *Proc. of Real-Time Systems Symposium*, pp. 1–12, 2015.

[18] B. B. Brandenburg and J. H. Anderson, "A comparison of the m-pcp, d-pcp, and fmlp on litmusrt," in *Principles of Distributed Systems*, pp. 105–124, 2008.

[19] K. Lakshmanan, D. de Niz, and R. Rajkumar, "Mixed-criticality task synchronization in zero-slack scheduling," in *Proc. of the Real-Time and Embedded Technology and Applications Symposium*, pp. 47–56, 2011.

[20] Q. Zhao, Z. Gu, and H. Zeng, "Hlc-pcp: A resource synchronization protocol for certifiable mixed criticality scheduling," *IEEE Embedded Systems Letters*, vol. 6, no. 1, pp. 8–11, 2014.

[21] Q. Zhao, Z. Gu, and H. Zeng, "Resource synchronization and preemption thresholds within mixed-criticality scheduling," *ACM Transactions on Embedded Computing Systems*, vol. 14, no. 4, p. 81, 2015.

[22] A. Burns, "The application of the original priority ceiling protocol to mixed criticality systems," in *Proc. of International Conference on Real-Time Computing Systems and Applications*, pp. 7–11, 2013.

[23] Y. Cui, Y. Wang, Y. Chen, and Y. Shi, "Requester-based spin lock: A scalable and energy efficient locking scheme on multicore systems," *IEEE Transactions on Computers*, vol. 64, no. 1, pp. 166–179, 2015.

[24] S. Baruah, V. Bonifaci, G. D'angelo, H. Li, A. Marchetti-Spaccamela, S. Van Der Ster, and L. Stougie, "Preemptive uniprocessor scheduling of mixed-criticality sporadic task systems," *Journal of the ACM*, vol. 62, no. 2, p. 14, 2015.

[25] B. B. Brandenburg and J.-H. Anderson, "Optimality results for multiprocessor real-time locking," in *Proc. of the IEEE Real-Time Systems Symposium*, pp. 49–60, 2010.

[26] A. Burns and A. J. Wellings, "A schedulability compatible multiprocessor resource sharing protocol–mrsp," in *Proc. of the 25th Euromicro Conference on Real-Time Systems*, pp. 282–291, 2013.

[27] J.-J. Han, X. Wu, D. Zhu, H. Jin, L. T. Yang, and J.-L. Gaudiot, "Synchronization-aware energy management for vfi-based multicore real-time systems," *IEEE Trans. on Computers*, vol. 61, no. 12, pp. 1682–1696, 2012.

[28] J.-J. Han, D. Zhu, X. Wu, L. T. Yang, and H. Jin, "Multiprocessor real-time systems with shared resources: Utilization bound and mapping," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 11, pp. 2981–2991, 2014.

[29] P.-C. Hsiu, D.-N. Lee, and T.-W. Kuo, "Task synchronization and allocation for many-core real-time systems," in *Proc. of the ACM Int'l Conf. on Embedded software*, pp. 79–88, 2011.

[30] Y.-S. Chen, C.-Y. Yang, and T.-W. Kuo, "Energy-Efficient Task Synchronization for Real-Time Systems," *IEEE Transactions on Industrial Informatics*, vol. 6, no. 3, pp. 287–301, 2010.

[31] Y. Wang and M. Saksena, "Scheduling fixed-priority tasks with preemption threshold," in *Proc. of the Sixth International Conference on Real-Time Computing Systems and Applications*, pp. 328–335, 1999.

[32] B. B. Brandenburg, "A synchronous ipc protocol for predictable access to shared resources in mixed-criticality systems," in *Proc. of the IEEE Real-Time Systems Symposium*, pp. 196–206, 2014.

[33] A. Burns and R. I. Davis, "Mixed criticality systems-a review," in *Department of Computer Science, University of York, Tech. Rep.*, 2016.

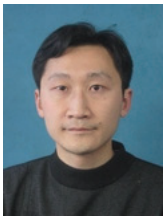
[34] S. K. Baruah, A. Burns, and R. I. Davis, "Response-time analysis for mixed criticality systems," in *Proc. of the IEEEReal-Time Systems Symposium*, pp. 34–43, 2011.

[35] H. Li and S. Baruah, "Global mixed-criticality scheduling on multiprocessors," in *Proc. of 24th Euromicro Conference on Real-Time Systems*, pp. 166–175, 2012.

[36] J. Ras and A. M. Cheng, "An evaluation of the dynamic and static multiprocessor priority ceiling protocol and the multiprocessor stack resource policy in an smp system," in *Real-Time and Embedded Technology and Applications Symposium, 2009. RTAS 2009. 15th IEEE*, pp. 13–22, 2009.

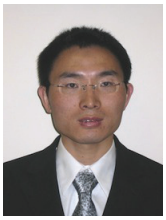
[37] Y.-K. Kwok and I. Ahmad, "Static scheduling algorithms for allocating directed task graphs to multiprocessors," *ACM Computing Surveys*, vol. 31, no. 4, pp. 406–471, 1999.

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

**Jian-Jun Han** received the PhD degree in computer science and engineering from Huazhong University of Science and Technology (HUST), in 2005. He is now an Associate Professor at the School of Computer Science and Technology in HUST. He worked at the University of California, Irvine as a visiting scholar between 2008 and 2009, and at the Seoul National University between 2009 and 2010. His research interests include real-time systems, parallel processing and green computing.

**Xin Tao** is now a Master Candidate in the School of Computer Science and Technology at HUST. His current research interests include real-time scheduling algorithm, embedded systems and operating systems.



**Dakai Zhu** received the PhD degree in Computer Science from University of Pittsburgh in 2004. He is currently an Associate Professor in the Department of Computer Science at the University of Texas at San Antonio. His research interests include real-time systems, power aware computing and fault-tolerant systems. He was a recipient of the US National Science Foundation (NSF) Faculty Early Career Development (CAREER) Award in 2010.

**Laurence T. Yang** received the PhD degree in computer science from the University of Victoria, BC, Canada. He is currently a Professor at School of Computer Science and Technology, Huazhong University of Science and Technology, China and at Department of Computer Science, St. Francis Xavier University, Canada. His current research interests include parallel and distributed computing, embedded and ubiquitous/pervasive computing.