Precise Scheduling Mixed-Criticality Gang Tasks with Reserved Processors

Tianning She
Department of Computer Science
Texas State University
San Marcos, TX, USA
t_s374@txstate.edu

Zhishan Guo
Department of Computer Science
North Carolina State University
Raleigh, NC, USA
zguo32@ncsu.edu

Kecheng Yang
Department of Computer Science
Texas State University
San Marcos, TX, USA
yangk@txstate.edu

***Abstract*—To mitigate the analytic pessimism that is often necessary to provide the worst-case guarantees for real-time systems, mixed-criticality (MC) scheduling has been proposed, where a task parameter may be associated with multiple estimates corresponding to multiple system runtime modes. While a large body of work on MC scheduling is directed at dropping or degrading low-critical tasks at the mode switch, a recent model, called precise MC scheduling, aims at preserving the full execution of all tasks instead. In precise MC scheduling, the additional workload due to high-critical tasks at the mode switch should be dealt with by increasing the capability of the processing platform. In this paper, we investigate the problem of precise MC scheduling of gang tasks, which may require simultaneously occupying multiple processors to commence any execution. In particular, we focus on the global earliest-deadline-first with virtual deadlines (GEDF-VD) scheduling. We derive a sufficient schedulability test for precise scheduling MC gang tasks by GEDF-VD. By the schedulability test, we also present an analysis of how many processors can be safely reserved in a given system.**

***Index Terms*—mixed-criticality tasks, precise scheduling, gang tasks, virtual deadlines.**

I. Introduction

In order to provide the worst-case guarantees for real-time systems, task parameters, such as the execution time, for schedulability analysis are usually estimated with significant pessimism, which results in less efficient runtime performance. Mixed-criticality (MC) scheduling has been proposed to mitigate such pessimism, where the worst-case execution time (WCET) of a task may be associated with multiple estimates corresponding to multiple system runtime modes.

In particular, the research attention in MC scheduling has often focused on the two-mode setting, where a mode switch is triggered by a high-critical task overrunning its smaller estimate. In the majority of existing work on MC scheduling, once a mode switch is triggered by such one or more high-critical tasks, the low-critical tasks are (fully or partially) sacrificed in order to accommodate the additional workload by those high-critical tasks. More recently, a new direction, called precise MC scheduling, has been proposed and investigated [8]. In the precise MC scheduling paradigm, after a mode switch, low-critical tasks continue to execute up to their WCET estimates as they do before the mode switch. In the meanwhile, the additional workload by the high-critical tasks are supposed to be covered by increasing the executing platform capacity, *e.g.*, boosting the speed of the processor [8].

Besides processor speed, increasing the number of available processors on a multiprocessor platform is another natural way for increasing capacity. In other words, in the normal mode, a certain number of processors are reserved and only the remaining number of processors are deemed available for the real-time tasks of our interest. Upon a mode switch, these reserved processors become available and dedicated to the real-time tasks so that the executing platform capacity is boosted from the real-time tasks' point of view. Such reserved processors in the normal mode could serve for several different purposes, such as to enter a sleep state for power and energy benefits, to be devoted to non-real-time tasks for spatial separation between real-time and non-real-time tasks (the latter will be dropped or deprioritized upon a mode switch), *etc.*, depending on the scenarios and needs of the system.

In this precise MC scheduling scheme with reserved processors, it is the number of processors that varies. Compared to traditional sequential tasks, *parallel* tasks are naturally more tuned to this particular system parameter. In this paper, we focus on one particular kind of parallel tasks, namely the (rigid) gang task model, where a task may *require* simultaneously occupying multiple processor to commence any execution. The real-time scheduling of gang tasks have received some attention [24, 16, 17, 14]. It has also been studied in the context of MC scheduling [9] but only in the conventional sense (*i.e.*, dropping tasks upon a mode switch).

Contributions. In this paper, we present the first study on the precise MC scheduling of gang tasks. With a certain number of reserved processors that are hidden from the real-time tasks in the normal mode but are able to fully dedicate to real-time tasks upon a mode switch, we formally define a specific scheduling problem in this direction. We then devise a virtual-deadline based scheduling algorithm to address this problem, and present a schedulability test for this algorithm. By the schedulability test, we also present an analysis of how many processors can be safely reserved in a given system.

Organization. In the rest of this paper, we introduce our system model and problem statement (Sec. II), present a common deadline-based scheduling algorithm for general gang tasks and its variant by virtual deadlines for MC gang tasks

(Sec. III), provide a proof and analysis for our schedulability test (Sec. IV), briefly survey related work (Sec. V), and conclude (Sec. VI).

II. System Model and Problem Statement

We consider the sporadic gang task model, which differs from the conventional sporadic task model by allowing a single task to simultaneously occupy multiple processors for execution. Furthermore, the allowed parallelism can be categorized as follows [17]: a job (invocation of a task) is

- *rigid* if the number of processors assigned to this job is specified externally to the scheduler a priori, and does not change throughout its execution;
- *moldable* if the number of processors assigned to this job is determined by the scheduler, and does not change throughout its execution
- *malleable* if the number of processors assigned to this job can be changed by the scheduler during the job's execution.

In this paper, we focus on the *rigid* parallelism model.

Specifically, we consider a system consisting of n implicit-deadline sporadic MC gang tasks $\mathcal{T} = \{\tau_1, \tau_2, \cdots, \tau_n\}$, where each task τ_i has a rigid parallelism of m_i processors. Each task τ_i is invoked recurrently with a minimum separation of T_i time units. Each invocation is called a *job* of τ_i, and we use $\tau_{i,j}$ to denote the j^{th} job of τ_i. T_i is called the *period* of τ_i, and we restrict our attention to implicit deadlines. In other words, T_i is also the *relative deadline* for each task τ_i, and every job of τ_i has an absolute deadline T_i time units after its release. The WCET of each task τ_i is estimated a two criticality levels: a low-criticality (L-) estimate C_i^L and a high-criticality (H-) estimate C_i^H, where it is assumed that $\forall i, 0 < C_i^L \leq C_i^H \leq T_i$. Furthermore, if $C_i^L = C_i^H$ for task τ_i so that τ_i *cannot* trigger a *mode switch* as to be described next, then we say τ_i is a LO-task; by contrast, if $C_i^L < C_i^H$ for task τ_i so that τ_i *could* trigger a *mode switch* as to be described next, then we say τ_i is a HI-task. We denote the set of LO-tasks (HI-tasks, respectively) by $\mathcal{T}_{\text{LO}}$ ($\mathcal{T}_{\text{HI}}$, respectively). We also refer to a job of a LO-task (HI-task, respectively) as LO-job (HI-job, respectively) for short. In summary, an implicit-deadline MC sporadic gang task τ_i is specified by a 4-tuple (C_i^L, C_i^H, m_i, T_i).

Reserving processors and mode switch. We consider a multiprocessor platform consisting of M^H identical processors, each of which has a normalized speed 1.0. In the runtime, if the L-estimates are respected, *i.e.*, *all* jobs are finished within their L-WCETs, then we say the system is in *L-mode*; if the L-estimates are exceeded, *i.e.*, *some* jobs need to execute beyond their L-WCETs and up to their H-WCETs, then we say the system is in *H-mode*. Note that the H-estimates are assumed to be always respected. In other words, any job that has cumulatively executed for its H-WCET, *i.e.*, C_i^H, yet still not completed, is considered as *erroneous* and would be terminated immediately. That is, only HI-tasks, for which $C_i^L < C_i^H$, could trigger a mode switch. The system begins with L-mode and the amount of execution completed for each job is being monitored during runtime. If any job has cumulatively executed for its L-WCET, *i.e.*, C_i^L, but still requires further execution, then the system is immediately notified and switched to H-mode. The system can recover to L-mode once *all* processors become idle. We require that only M^L, where $M^L < M^H$, processors are used to actively execute tasks in $\mathcal{T}$ in L-mode, while the remaining $M^\delta = (M^H - M^L)$ processors are reserved. Nonetheless, once the system is switched to H-mode, all M^H processors are devoted to execute tasks in $\mathcal{T}$.

Note that, in contrast to the majority of existing works on MC scheduling, in this work *no task is entirely or partially dropped* upon a mode switch, and every job must meet its absolute deadline in any system mode. The difference between the two WCET estimates upon mode switch, *i.e.*, $C_i^H - C_i^L$, is compensated by the additional M^δ active processors.

In this paper, we assume that the preemption and migration overheads, *e.g.*, due to memory interference, are negligible. Or, equivalently, we assume these overheads are pessimistically taken into account in the WCET estimates.

Moreover, we denote the utilization of a task τ_i in L- and H-modes, respectively, by

$$u_i^L = \frac{C_i^L \cdot m_i}{T_i} \quad \text{and} \quad u_i^H = \frac{C_i^H \cdot m_i}{T_i}.$$

Since $C_i^L = C_i^H$ holds for every LO-task, it also holds $u_i^L = u_i^H$ for such task. We further denote the total utilization of the set of LO-tasks and the set of HI-tasks in L- and H-modes, respectively, by

$$U_{\text{LO}} = \sum_{\tau_i \in \mathcal{T}_{\text{LO}}} u_i^L = \sum_{\tau_i \in \mathcal{T}_{\text{LO}}} u_i^H,$$

$$U_{\text{HI}}^L = \sum_{\tau_i \in \mathcal{T}_{\text{HI}}} u_i^L, \text{ and } U_{\text{HI}}^H = \sum_{\tau_i \in \mathcal{T}_{\text{HI}}} u_i^H.$$

We further denote the total utilization of all tasks in L- and H-modes, respectively, by

$$U^L = \sum_i u_i^L = U_{\text{LO}} + U_{\text{HI}}^L \quad \text{and} \quad U^H = \sum_i u_i^H = U_{\text{LO}} + U_{\text{HI}}^H.$$

Problem Statement. We address the problem of scheduling the MC rigid Gang tasks on M^H unit-speed processors to meet all deadlines in *all* scenarios with the potential of reserving M^δ processors, where $M^\delta = M^H - M^L > 0$. We say the system is *precise-MC schedulable* if all deadlines are guaranteed to be met and the following constraints are respected.

- Tasks in τ only execute on M^L processors if *all* jobs finish within C_i^L time units of execution;
- Tasks in τ may execute on all the M^H processors if a *any* job (of a HI-task) executes for more than C_i^L time units (yet finishes within C_i^H time units of execution).

III. Scheduling MC Gang Tasks

In this section, we propose an earliest-deadline-first (EDF) based scheduling algorithm to address the precise MC scheduling problem of gang tasks, which has been formalized in the prior section.

Algorithm 1: Selecting Jobs to Schedule under GEDF

input : Ready(t), which is the ready job set at time t
output: Sched(t), which is the scheduled job set at time t

Sched$(t) \leftarrow \emptyset$
for each $\tau_{i,j} \in$ Ready(t) in deadline increasing order **do**
  if $\sum_{\tau_{k,\ell} \in \text{Sched}(t)} m_k \leq M - m_i$ **then**
    Sched$(t) \leftarrow$ Sched$(t) \cup \{\tau_{i,j}\}$
  end
end

A. GEDF *Scheduling for Gang Tasks*

We consider the *preemptive* global EDF (GEDF) scheduling algorithm for *ordinary (non-MC)* gang tasks as follows[1]:

Definition 1. Under GEDF scheduling, the priority of each job is determined by its deadline — the earlier the deadline, the higher the priority. We also assume deadline ties are broken arbitrarily but consistently, and therefore there is no *priority* ties while *deadline* ties may exist. Letting Ready(t) denote the set of *ready* jobs at an arbitrary time instant t, the set of jobs Sched(t) being scheduled at time t is determined by Algorithm 1. m_k is the degree of parallelism of τ_k, which has a job in Sched(t). Please note that, in practice, Algorithm 1 does not need to be evaluated at every time instant but only needs to be invoked when a job is completed and when a new job is released.

Please note that the GEDF scheduling cannot be defined by simply considering the summation of parallelism of all ready jobs with higher priorities, but rather it needs to go through Algorithm 1. This is because some higher-priority (but not the highest) ready jobs may not be scheduled due to lack of available processors while some lower-priority ones with less parallelism may actually be scheduled.

Furthermore, an important parameter Δ_i for a gang task τ_i under GEDF scheduling was introduced in [14] and is defined as follows.

Definition 2. Let Δ_i denote the maximum possible number of idle processors at any time during τ_i's non-executing intervals in which τ_i has pending jobs but does not execute. In other words, if τ_i is not being executing but has pending jobs, the number of idle processors is at most Δ_i.

Clearly, $(m_i - 1)$ would be a *safe* upperbound on Δ_i. Nonetheless, a dynamic programming algorithm has been introduced in [14] to calculate Δ_i in a more accurate manner. This Δ_i identification algorithm runs in polynomial time with a time complexity of $\mathcal{O}(M^2 \cdot n)$, where M is the total number of available processors and n is the number of gang tasks.

With pre-calculating the Δ_i parameter for every task prior to runtime for any given (non-MC) gang task system, a sufficient schedulability test has been proven in [14] as follows.

[1]This formal description of GEDF for gang tasks was introduced in [15].

Theorem 1 (Theorem 2 in [14])**.** *A (non-MC) gang task system, where each task is specified by (C_i, m_i, T_i), is schedulable on M identical processors by* GEDF*, if*

$$\forall i, U_{\text{SUM}} \leq (M - \Delta_i)(1 - \frac{u_i}{m_i}) + u_i \tag{1}$$

where

$$u_i = \frac{C_i}{T_i} \quad \textit{and} \quad U_{\text{SUM}} = \sum_i u_i.$$

B. *Algorithm* GEDF-VD *for MC Gang Tasks*

With the prior work on GEDF scheduling of non-MC gang tasks as discussed above, we introduce a virtual-deadline-based scheduling algorithm, GEDF-VD, to address the precise MC scheduling problem considered in this paper.

Under GEDF-VD, a system wide constant x is selected such that $0 < x < 1$. Each task is associated to a *relative virtual deadline* of $x \cdot T_i$, in addition to the *relative actual deadline* T_i. That is, every job of task τ_i has a virtual deadline at $x \cdot T_i$ time units after its arrival time and has an actual deadline at T_i time units after its arrival. Note that, because under precise MC scheduling LO-tasks are not dropped in the H-mode, the virtual-deadline setting does also apply to LO-tasks as well as HI-tasks. Then, in the runtime, GEDF is applied to schedule all tasks by taking virtual deadlines as deadlines in the L-mode and taking actual deadlines as deadlines in the H-mode, respectively.

Recall that for non-MC gang task set, we have a Δ_i parameter that can be obtained before runtime. The Δ_i identification algorithm from [14] needs to take the required parallelism m_i for every task and the total number of available processors M as inputs. In the problem considered in this paper, although m_i remains the same in the L- and H-modes for every task τ_i, the total number of available processors does differ in the two modes, being M^L and M^H in L- and H-modes, respectively. Therefore, we need to apply the Δ_i identification algorithm twice, and then for each task τ_i, we obtain Δ_i^L and Δ_i^H for L- and H-modes, respectively.

Having Δ_i^L and Δ_i^H for every task, we claim the following sufficient schedulability test for GEDF-VD and this test is to be proven next in Sec. IV.A.

A set of precise MC gang tasks with implicit deadlines are schedulable by GEDF-VD using at most M^L processors in the L-mode and using at most M^H processors in the H-mode, if

$$K^L + K^H \leq 1,$$

where

$$K^L = \max_i \left\{ \frac{m_i U^L + (M^L - \Delta_i^L - m_i) u_i^L}{m_i \cdot (M^L - \Delta_i^L)} \right\}$$

$$K^H = \max_i \left\{ \frac{m_i U^H + (M^H - \Delta_i^H - m_i) u_i^H}{m_i \cdot (M^H - \Delta_i^H)} \right\},$$

so that scaling factor x for setting virtual deadlines in GEDF-VD can be (arbitrarily) chosen from the range

$$[K^L, 1 - K^H].$$

IV. Precise-MC Schedulability Analysis

In this section, we first prove the correctness of the schedulability presented in the prior section, with given M^L and M^H (and therefore given Δ_i^L and Δ_i^H). Then, we provide a method to obtain a safe lower bound on M^L that guarantees the schedulability for given task system and given M^H.

A. Schedulability Test

To derive the schedulability test, we consider the schedulability in L- and H-modes, respectively, in the following two lemmas. We first derive a sufficient schedulability condition for L-mode (Lem. 1). Then, assuming the schedulability in L-mode, we derive a sufficient schedulability condition for H-mode (Lem. 2). They together result in Thm. 2.

Lemma 1. *Under* GEDF-VD *scheduling, all (*LO- *and* HI-*) tasks must meet their virtual deadlines in L-mode, if*

$$\forall i, x \geq \frac{m_i U^L + (M^L - \Delta_i^L - m_i)u_i^L}{m_i \cdot (M^L - \Delta_i^L)}.$$

Proof. In L-mode, tasks are scheduled by their virtual deadlines. By treating the relative virtual deadline as both the relative deadline and the period, every task τ_i in L-mode can be viewed as a (sporadic, implicit-deadline) non-MC gang task specified as (C_i^L, m_i, xT_i). Under this view, the utilization of each task τ_i is

$$\frac{C_i^L}{xT_i} = \frac{u_i^L}{x},$$

and the total utilization is

$$\sum_i \frac{C_i^L}{xT_i} = \frac{1}{x} \cdot \sum_i \frac{C_i^L}{T_i} = \frac{U^L}{x}.$$

Therefore, by Thm. 1, these non-MC gang tasks must meet their deadlines under GEDF, *i.e.*, original MC-gang tasks must meet their virtual deadlines in L-mode, if

$$\forall i, \frac{U^L}{x} \leq (M^L - \Delta_i^L)(1 - \frac{\frac{u_i^L}{x}}{m_i}) + \frac{u_i^L}{x} \tag{2}$$

Given the facts that $0 < x < 1$, $m_i > 0$, and $\Delta_i^L < M^L$,

$$\begin{aligned}
(2) \Leftrightarrow & \forall i, \frac{U^L}{x} \leq M^L - \Delta_i^L - (\frac{M^L - \Delta_i^L}{m_i} - 1) \cdot \frac{u_i^L}{x} \\
\Leftrightarrow & \forall i, \frac{m_i U^L + (M^L - \Delta_i^L - m_i)u_i^L}{m_i x} \leq M^L - \Delta_i^L \\
\Leftrightarrow & \forall i, x \geq \frac{m_i U^L + (M^L - \Delta_i^L - m_i)u_i^L}{m_i \cdot (M^L - \Delta_i^L)}.
\end{aligned}$$

Thus, the lemma follows. □

Lemma 2. *Under* GEDF-VD *scheduling, assuming all virtual deadlines are met in L-mode, all (*LO- *and* HI-*) tasks must meet their actual deadlines in H-mode, if*

$$\forall i, x \leq 1 - \frac{m_i U^H + (M^H - \Delta_i^H - m_i)u_i^H}{m_i \cdot (M^H - \Delta_i^H)}.$$

Proof. By assuming all virtual deadlines are met in L-mode, it follows that the first job that reaches its virtual deadline but has not completed must trigger a mode switch. As a result, any job of τ_i that is released in L-mode has not completed by the mode switch must have its actual deadline (as well as next job release of τ_i) at least $(1-x)T_i$ time units after the mode-switch time instant. In the meanwhile, every job of τ_i cannot execute for more than C_i^H time units in any circumstance and all subsequent releases in H-mode must separate by $T_i > (1-x)T_i$ time units, according to the task model. So, by treating any unfinished job at the mode switch as a new job release at the mode-switch time instant, every task τ_i in L-mode can be viewed as a (sporadic, implicit-deadline) non-MC gang task specified as $(C_i^H, m_i, (1-x)T_i)$. Under this view, the utilization of each task τ_i is

$$\frac{C_i^H}{(1-x)T_i} = \frac{u_i^H}{1-x},$$

and the total utilization is

$$\sum_i \frac{C_i^H}{(1-x)T_i} = \frac{1}{1-x} \cdot \sum_i \frac{C_i^H}{T_i} = \frac{U^H}{1-x}.$$

Therefore, by Thm. 1, these non-MC gang tasks must meet their deadlines under GEDF, *i.e.*, original MC-gang tasks must meet their actual deadlines in H-mode, if

$$\forall i, \frac{U^H}{1-x} \leq (M^L - \Delta_i^L)(1 - \frac{\frac{u_i^H}{1-x}}{m_i}) + \frac{u_i^H}{1-x} \tag{3}$$

Given the facts that $0 < x < 1$, $m_i > 0$, and $\Delta_i^H < M^H$,

$$\begin{aligned}
(3) \Leftrightarrow & \forall i, \frac{U^H}{1-x} \leq M^H - \Delta_i^H - (\frac{M^H - \Delta_i^H}{m_i} - 1) \cdot \frac{u_i^H}{1-x} \\
\Leftrightarrow & \forall i, \frac{m_i U^H + (M^H - \Delta_i^H - m_i)u_i^H}{m_i(1-x)} \leq M^H - \Delta_i^H \\
\Leftrightarrow & \forall i, 1 - x \geq \frac{m_i U^H + (M^H - \Delta_i^H - m_i)u_i^H}{m_i \cdot (M^H - \Delta_i^H)} \\
\Leftrightarrow & \forall i, x \leq 1 - \frac{m_i U^H + (M^H - \Delta_i^H - m_i)u_i^H}{m_i \cdot (M^H - \Delta_i^H)}.
\end{aligned}$$

Thus, the lemma follows. □

Theorem 2. *An MC gang task system is precise-MC schedulable under* GEDF-VD *on M^H processors with $M^\delta = (M^H - M^L)$ processors reserved in L-mode, if*

$$K^L + K^H \leq 1,$$

where

$$K^L = \max_i \left\{ \frac{m_i U^L + (M^L - \Delta_i^L - m_i)u_i^L}{m_i \cdot (M^L - \Delta_i^L)} \right\}$$

$$K^H = \max_i \left\{ \frac{m_i U^H + (M^H - \Delta_i^H - m_i)u_i^H}{m_i \cdot (M^H - \Delta_i^H)} \right\}.$$

Proof. $K^L + K^H \leq 1$ implies that $[K^L, 1 - K^H]$ is not an empty set. Also, it is evident that both $K^L > 0$ and $K^H >$

0, so $[K^L, 1-K^H] \subset (0,1)$. Therefore, a valid x can be (arbitrarily) selected from $[K^L, 1-K^H]$. Note that

$$x \geq K^L \Rightarrow \forall i, x \geq \frac{m_i U^L + (M^L - \Delta_i^L - m_i)u_i^L}{m_i \cdot (M^L - \Delta_i^L)} \quad \text{and}$$

$$x \leq 1-K^H \Rightarrow \forall i, x \leq 1 - \frac{m_i U^H + (M^H - \Delta_i^H - m_i)u_i^H}{m_i \cdot (M^H - \Delta_i^H)}.$$

By Lem. 1 and Lem. 2, the theorem follows. □

B. A Safe Lower Bound on M^L for Schedulability

We have already established a schedulability test for a given task system with given M^L and given M^H. Nonetheless, the value of M^L may not always be fixed in a prior but its proper setting may be a question for the system designer, where the system workload specification ($\mathcal{T}$) and the platform maximum capacity (M^H) are provided.

Under such a setting, K^H can still be obtained *a priori*, as $\mathcal{T}$ and M^H (and therefore, Δ_i^H) are given. Nonetheless, to evaluate the schedulability condition, *i.e.*, $K^L \leq 1 - K^H$, it is not only M^L that varies— each Δ_i^L could also be different under different M^L, and it is not a closed-form representation by M^L. One way to find an M^L that guarantees schedulability is to enumerate all possible values of M^L and to apply the schedulability test for each case. Alternatively, the following theorem directly provides a safe lower bound on M^L that guarantees the schedulability for certain systems.

Theorem 3. *For any MC gang task system where*

$$\forall i, (1-K^H)m_i > u_i, \tag{4}$$

it must be precise-MC schedulable under GEDF-VD*, if*

$$M^L \geq \max_i \left\{ \frac{m_i(U^L - u_i^L)}{(1-K^H)m_i - u_i^L} + m_i - 1 \right\}. \tag{5}$$

Proof. Our goal is to show that the lower bound on M^L specified in this theorem is sufficient to imply $K^L + K^H \leq 1$, which is the schedulability test.

From (5), it directly follows that

$$\begin{aligned} &\forall i, M^L \geq \frac{m_i(U^L - u_i^L)}{(1-K^H)m_i - u_i^L} + m_i - 1 \\ \Rightarrow &\forall i, M^L - (m_i - 1) \geq \frac{m_i(U^L - u_i^L)}{(1-K^H)m_i - u_i^L} \end{aligned} \tag{6}$$

While every Δ_i^L may vary as M^L varies, we note that $\forall i, \Delta_i^L \leq m_i - 1$ holds by definition in any case. Therefore, regardless the specific value of M^L, we always have

$$\begin{aligned} (6) &\Rightarrow \forall i, M^L - \Delta_i \geq \frac{m_i(U^L - u_i)}{(1-K^H)m_i - u_i^L} \\ &\overset{\text{by (4)}}{\Rightarrow} \forall i, (1-K^H)m_i \geq \frac{m_i(U^L - u_i^L)}{M^L - \Delta_i} + u_i^L \\ &\Rightarrow \forall i, (1-K^H) \geq \frac{m_i U^L + (M^L - \Delta_i^L - m_i)u_i^L}{m_i \cdot (M^L - \Delta_i^L)} \\ &\Rightarrow (1-K^H) \geq K^L, \end{aligned}$$

where the last step is by the definition of K^L and directly implies $K^L + K^H \leq 1$. The theorem follows. □

V. Related Work

Since it was introduced by Vestal [34], MC tasks and their scheduling have attracted a huge amount of interest in the real-time systems research community. (Please see [12] for a comprehensive survey on this topic.) Initially, most works were directed to scenarios where all low-critical tasks are completely dropped if any high-critical task behaves its worst case. More recently, this over-sacrificing was criticized, and gradual degradation of low-critical tasks was investigated. To provide degraded service, the imprecise MC model [11] was proposed, where the execution of low-critical tasks is reduced but not dropped even in the worst case. Several subsequent works [3, 11, 20, 22, 23, 27] explored various definitions of this execution reduction. To eliminate such reduction, the problem of precise MC scheduling was proposed and investigated on varying-speed uniprocessors [8, 35] and multiprocessors [33]. Such varying-speed processors are equipped with a capacity of *dynamic voltage and frequency scaling* (DVFS) for the purpose of energy efficiency [21]. However, DVFS is not effective in reducing static/leakage power consumption. Compared to DVFS, *dynamic power management* (DPM) and deep sleep modes can lead to significant energy conservation resulted from the power-down of a number of system components [5, 4]. [32] proposed precise MC scheduling of sequential tasks on multiprocessors that a part of processors in the system can be turned into sleep modes in typical scenarios while fully exploited under worst-case scenarios.

Besides sequential tasks, the problem of scheduling real-time parallel tasks has been investigated. Several works focus on parallel task models that are related to the gang task model, including periodic multi-thread task models [29, 31, 13], the synchronous task model [1], and DAG task models [10, 25, 19, 7, 18]. In these models, Parallel threads of a task can be independently considered and scheduled. By contrast, they must simultaneously occupy a set of processors to execute under the gang task model [17, 14, 24]. Goossens et al. considered the rigid gang task model in [16], and Berten et al. proposed the moldable gang scheduling in [6].

Upon MC scheduling, few works have been proposed for parallel task models. Liu et al. [28] proposed the MC scheduling of synchronous task model, while the MC scheduling for DAG task models is investigated in [2] and [26]. Rambo et al. [30] proposed a replica-aware co-scheduling approach for mixed-critical systems. For the MC gang task scheduling, [9] presented an approach combining Global Earliest Deadline First (GEDF) and Earliest Deadline First with Virtual Deadline (EDF-VD). Unlike these works, we consider precise MC scheduling of the gang task model that provides full service for low-critical gang tasks.

VI. Conclusion

In many conventional MC scheduling problems, LO-tasks may be dropped or degraded when a mode switch to H-mode is triggered. In this work, we investigated the precise MC scheduling, where LO-tasks preserve their execution estimates in H-mode. Upon a mode switch to H-mode, a certain number

of processors that are reserved in L-mode now become fully available and dedicated to the real-time tasks. Our focus in this paper was on rigid gang tasks, each of which may require multiple processors to commence any execution. We presented our scheduling algorithm EDF-VD for the problem of precise scheduling MC gang tasks and provided a schedulability test for EDF-VD as well as a proof for its correctness. In addition, we also analyzed a safe lower bound on the number of non-reserved processors in L-mode for guaranteeing the schedulability of a given system. This, from the other hand, implies how many processors can be safely reserved in L-mode without jeopardizing the schedulability.

ACKNOWLEDGMENT

This work is supported in part by NSF grants CCF-2028481, CNS-2104181, a start-up grant from the North Carolina State University, and start-up and REP grants from Texas State University.

REFERENCES

[1] Björn Andersson and Dionisio de Niz. Analyzing global-EDF for multiprocessor scheduling of parallel tasks. In *International Conference On Principles Of Distributed Systems*, pages 16–30. Springer, 2012.

[2] Sanjoy Baruah. The federated scheduling of systems of mixed-criticality sporadic dag tasks. In *2016 IEEE Real-Time Systems Symposium (RTSS)*, pages 227–236. IEEE, 2016.

[3] Sanjoy Baruah, Alan Burns, and Zhishan Guo. Scheduling mixed-criticality systems to guarantee some service under all non-erroneous behaviors. In *2016 28th Euromicro Conference on Real-Time Systems (ECRTS)*, pages 131–138. IEEE, 2016.

[4] Luca Benini, Alessandro Bogliolo, and Giovanni De Micheli. A survey of design techniques for system-level dynamic power management. *IEEE transactions on very large scale integration (VLSI) systems*, 8(3):299–316, 2000.

[5] Luca Benini, Alessandro Bogliolo, Giuseppe A Paleologo, and Giovanni De Micheli. Policy optimization for dynamic power management. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 18(6):813–833, 1999.

[6] Vandy Berten, Pierre Courbin, and Joël Goossens. Gang fixed priority scheduling of periodic moldable real-time tasks. In *5th junior researcher workshop on real-time computing*, pages 9–12. Citeseer, 2011.

[7] Ashikahmed Bhuiyan, Zhishan Guo, Abusayeed Saifullah, Nan Guan, and Haoyi Xiong. Energy-efficient real-time scheduling of DAG tasks. *ACM Transactions on Embedded Computing Systems*, 17(5):84, 2018.

[8] Ashikahmed Bhuiyan, Sai Sruti, Zhishan Guo, and Kecheng Yang. Precise scheduling of mixed-criticality tasks by varying processor speed. In *Proceedings of the 27th International Conference on Real-Time Networks and Systems*, pages 123–132, 2019.

[9] Ashikahmed Bhuiyan, Kecheng Yang, Samsil Arefin, Abusayeed Saifullah, Nan Guan, and Zhishan Guo. Mixed-criticality multicore scheduling of real-time gang task systems. In *2019 IEEE Real-Time Systems Symposium (RTSS)*, pages 469–480. IEEE, 2019.

[10] Vincenzo Bonifaci, Alberto Marchetti-Spaccamela, Sebastian Stiller, and Andreas Wiese. Feasibility analysis in the sporadic DAG task model. In *2013 25th Euromicro conference on real-time systems*, pages 225–233. IEEE, 2013.

[11] Alan Burns and Sanjoy Baruah. Towards a more practical model for mixed criticality systems. In *1st WMC*, 2013.

[12] Alan Burns and Robert Davis. Mixed criticality systems-a review. *Dept. of Computer Science, University of York, Tech. Rep*, pages 1–81, 2019.

[13] Pierre Courbin, Irina Lupu, and Joël Goossens. Scheduling of hard real-time multi-phase multi-thread (mpmt) periodic tasks. *Real-time systems*, 49(2):239–266, 2013.

[14] Zheng Dong and Cong Liu. Analysis techniques for supporting hard real-time sporadic gang task systems. *Real-Time Systems*, 55(3):641–666, 2019.

[15] Zheng Dong, Kecheng Yang, Nathan Fisher, and Cong Liu. Tardiness bounds for sporadic gang tasks under preemptive global edf scheduling. *IEEE Transactions on Parallel and Distributed Systems*, 32(12):2867–2879, 2021.

[16] Joël Goossens and Vandy Berten. Gang ftp scheduling of periodic and parallel rigid real-time tasks. *arXiv preprint arXiv:1006.2617*, 2010.

[17] Joël Goossens and Pascal Richard. Optimal scheduling of periodic gang tasks. *Leibniz transactions on embedded systems*, 3(1):04–1, 2016.

[18] Zhishan Guo, Ashikahmed Bhuiyan, Di Liu, Aamir Khan, Abusayeed Saifullah, and Nan Guan. Energy-efficient real-time scheduling of DAGs on clustered multi-core platforms. In *2019 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 156–168. IEEE, 2019.

[19] Zhishan Guo, Ashikahmed Bhuiyan, Abusayeed Saifullah, Nan Guan, and Haoyi Xiong. Energy-efficient multi-core scheduling for real-time DAG tasks. In *29th Euromicro conference on real-time systems (ECRTS 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

[20] Zhishan Guo, Kecheng Yang, Sudharsan Vaidhun, Samsil Arefin, Sajal K Das, and Haoyi Xiong. Uniprocessor mixed-criticality scheduling with graceful degradation by completion rate. In *2018 IEEE Real-Time Systems Symposium (RTSS)*, pages 373–383. IEEE, 2018.

[21] Pengcheng Huang, Pratyush Kumar, Georgia Giannopoulou, and Lothar Thiele. Energy efficient DVFS scheduling for mixed-criticality systems. In *Proceedings of the 14th International Conference on Embedded Software, ACM*, page 11. ACM, 2014.

[22] Pengcheng Huang, Pratyush Kumar, Georgia Giannopoulou, and Lothar Thiele. Run and be safe: Mixed-criticality scheduling with temporary processor speedup. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2015*, pages 1329–1334. IEEE, 2015.

[23] Mathieu Jan, Lilia Zaourar, and Maurice Pitel. Maximizing the execution rate of low criticality tasks in mixed criticality system. *1st WMC*, 2013.

[24] Shinpei Kato and Yutaka Ishikawa. Gang EDF scheduling of parallel task systems. In *2009 30th IEEE Real-Time Systems Symposium*, pages 459–468. IEEE, 2009.

[25] Jing Li, Jian-Jia Chen, Kunal Agrawal, Chenyang Lu, Christopher D Gill, and Abusayeed Saifullah. Analysis of federated and global scheduling for parallel real-time tasks. In *ECRTS*, volume 14, pages 85–96, 2014.

[26] Jing Li, David Ferry, Shaurya Ahuja, Kunal Agrawal, Christopher Gill, and Chenyang Lu. Mixed-criticality federated scheduling for parallel real-time tasks. *Real-time systems*, 53(5):760–811, 2017.

[27] Di Liu, Jelena Spasic, Nan Guan, Gang Chen, Songran Liu, Todor Stefanov, and Wang Yi. Edf-vd scheduling of mixed-criticality systems with degraded quality guarantees. In *2016 IEEE Real-Time Systems Symposium (RTSS)*, pages 35–46. IEEE, 2016.

[28] Guangdong Liu, Ying Lu, Shige Wang, and Zonghua Gu. Partitioned multiprocessor scheduling of mixed-criticality parallel jobs. In *2014 IEEE 20th International Conference on Embedded and Real-Time Computing Systems and Applications*, pages 1–10. IEEE, 2014.

[29] Geoffrey Nelissen, Vandy Berten, Joël Goossens, and Dragomir Milojevic. Techniques optimizing the number of processors to schedule multi-threaded tasks. In *2012 24th Euromicro Conference on Real-Time Systems*, pages 321–330. IEEE, 2012.

[30] Eberle A Rambo and Rolf Ernst. Replica-aware co-scheduling for mixed-criticality. In *29th Euromicro Conference on Real-Time Systems (ECRTS 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

[31] Abusayeed Saifullah, Jing Li, Kunal Agrawal, Chenyang Lu, and Christopher Gill. Multi-core real-time scheduling for generalized parallel task models. *Real-Time Systems*, 49(4):404–435, 2013.

[32] Tianning She, Zhishan Guo, Qijun Gu, and Kecheng Yang. Reserving processors by precise scheduling of mixed-criticality tasks. In *2021 IEEE 27th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, pages 103–108. IEEE, 2021.

[33] Tianning She, Sudharsan Vaidhun, Qijun Gu, Sajal K Das, Zhishan Guo, and Kecheng Yang. Precise scheduling of mixed-criticality tasks on varying-speed multiprocessors. In *Proceedings of the 29th International Conference on Real-Time Networks and Systems*, 2021.

[34] Steve Vestal. Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance. In *28th IEEE International Real-Time Systems Symposium (RTSS)*, pages 239–243. IEEE, 2007.

[35] Kecheng Yang, Ashikahmed Bhuiyan, and Zhishan Guo. F2VD: Fluid rates to virtual deadlines for precise mixed-criticality scheduling on a varying-speed processor. In *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pages 1–9. IEEE, 2020.