Integrating Cache-Related Preemption Delay into GEDF Analysis for Multiprocessor Scheduling with On-Chip Cache

Ying Zhang, Zhishan Guo, Lingxiang Wang, Haoyi Xiong
Missouri University of Science and Technology
Department of Computer Science,
Rolla, MO 65409, USA
Email: {yzhg4, guozh, lwqk6, xiongha}@mst.edu

Zhenkai Zhang
Vanderbilt University
Institute for Software Integrated Systems
Nashville, TN 37212, USA
Email: zhenkai.zhang@vanderbilt.edu

***Abstract*—Most existing multiprocessor schedulability analysis assumes zero cost for preemptions and migrations. In order for those analysis to be correct, execution time estimations are often inflated by a certain (pessimistic) factor, leading to severe waste of computing resource.In this paper, a novel Global Earliest Deadline First (GEDF) schedulability test is proposed, where Cache-Related Preemption Delay (CRPD) is separately modeled and integrated. Specifically, multiple analyses for estimating CRPD bounds are conducted based on the refined estimation of the maximal number of preemptions, leading to tighter G-EDF schedulability tests. The experimental study is conducted to demonstrate the performance of the proposed methods.**

***Keywords*-Multiprocessor scheduling, Cache-Related Preemption Delay, Global Earliest Deadline First, schedulability analysis**

I. INTRODUCTION

With the rapid growth of Cyber-Physical Systems (CPS) and Internet of Things (IoT) [17], Multiprocessors [8] have been widely used in embedded real-time systems in the last decade. The tremendous computing power of multi-processors have featured the embedded real-time systems with higher capacity but lower cost. In such a trend, both hardware and software providers have started to support multiprocessors/multi-core processors in practical embedded real-time systems design. For example, as a leading micro-processor provider in real-time system, ARM has released its ARMv8-A structure with multi-core configurations, while a series of real-time operating systems, such as VxWorks, have been upgraded to fully support multi-core processors.

In terms of performance, the embedded real-time systems equipped with multiprocessors are capable to schedule a larger volume of concurrent tasks, while guaranteeing all responses (e.g., task completion) on time [11]. To understand the response time of embedded real-time system, a branch of studies [3] [4] [5] [18] have been done to analyze schedulability in multi-processor. As early as 1973, to analyze the performance of Earliest Deadline First (EDF) scheduling in the multiprocessor, Liu and Layland [18] studied a sufficient condition for guaranteeing schedulability of all tasks. Then, to derived the maximum execution (time) requirement for each task set, Baker [3] [4] proposed a Global EDF (GEDF) schedulability test. Later in 2007, Baruah [5] and bertogna *et al*. [6] improved GEDF test and developed a new schedulability test. Most recently, in 2015, Sun *et al*. [23] proposed new schedulability test through response time analysis for GEDF.

However, existing studies on multiprocessors/multi-core processors rarely take the delay caused by *preempting shared resource* into consideration. Most existing schedulability analyses are based on certain unrealistic assumptions; e.g., zero time cost for preempting a shared resource. Note that the preemptions of shared resources commonly exist in the multiprocessor/multi-core systems and could cause significant performance degradation (e.g., missing deadlines) in the worst case scenarios [2] [9] [10]. One common way is to multiply the worst case execution time parameters by a certain factor to cover potential delays cased by preempting shared resource – this is often over pessimistic [22].

Related Work. Among a wide range of delay caused by shared resources such as bus and main memory [10] [14] [16], the Cache-Related Preemption Delay (CRPD) [15] [16] [20] is a crucial factor of schedulability guarantee in multiprocessor systems, while CRPD is usually overestimated under Earliest Deadline First (EDF) scheduling settings [3] [4]. In 2007, Ju *et al*. [13] integrated the CRPD into EDF schedulability analysis in *uniprocessor* settings, where they took all possible direct preemptions of a single job into account. Following Ju's attempts, Lunniss *et al*. [19] proposed an extended CRPD analysis for EDF, where they leveraged ECB-union multiset approach and UCB-union multiset approach to bound the CRPD. This result provided a tighter bound of CRPD compared with the work of Ju *et al* [13].

Unfortunately, the aforementioned existing work only analyzes the CRPD for EDF on *uniprocessor* platforms. The upper bound of CRPD is yet not known under EDF scheduling in the multiprocessor embedded real-time systems.

In this paper, we study the schedulability of EDF on multiprocessor systems taking into account the CRPD and derive a tight bound of CRPD under such settings. We propose a

2324-9013/17 $31.00 © 2017 IEEE
DOI 10.1109/Trustcom/BigDataSE/ICESS.2017.317

novel CRPD analytical approach that extends the existing the state of the art of CRPD analysis [19] [13] to GEDF scheduling. Specifically, while existing works [13] [19] assumes that each released job of tasks causes a preemption of shared cache to the job in execution, therefore all cache blocks are inferred densely in an uniprocessor, our work leverages the nature of the *sparse interferences* between cache blocks distributed on multi-cores/multi-processors [21].

Please note that the estimation of preemption times on uniprocessor may not be accurate under multiprocessor settings, as the actual number of preemption times on multiprocessor is lower than the uniprocessor case[1].

Organization and Contribution. In this paper, we propose a novel approach to derives a tighter upper bound of CRPD under GEDF scheduling on the multiprocessor platform. At Section II, We present the System Model and the notations used in this paper. Based on the models and notations, we introduce GEDF-CRPD Test – a new GEDF schedulability test on multiprocessor for CRPD analysis in Section III. With the new test, in Section IV, we propose an three-step approach that first condenses the multiset of interfered cache blocks, then estimates the maximal number of preemption times on multiprocessors, and further bounds the CRPD via our GEDF-CRPD Test. In Section V, we compare our method to the existing approaches; the experimental results show that our method converges to the Demand Bound Function (DBF) with a tighter margin than other methods. Section VI concludes the main contributions for this paper.

To the best of our knowledge, this is the **first** work that analyzes *the multiprocessor CRPD upper bound under global EDF scheduling*, by addressing interference cache blocks on multiprocessors and refined estimation of maximum preemption occurrences issues.

II. System Model and Notations

In order to analyze the CRPD in GEDF schedulability test, we first describes the system model, terminology, and notations used in the rest of the paper.

We consider a multicore system which has a fixed number of processors shared an on-chip one-level cache. Specifically, these processors do not have any private cache, as demonstrated in figure 1. Henceforth, no migration delay for tasks will be considered due to *no partitioning and no private cache*.

We assume a multiprocessor system with m processors running a predefined sporadic task set under GEDF scheduling, and the total number of tasks $n \gg m$. Each task τ_i defined as a 3-tuple $\{C_i, D_i, T_i\}$:

[1]Consider the following settings as an intuitive example demonstrating *sparse interference*: all tasks are executed on a single uniprocessor (with a single cache), the cache is shared by all tasks and the preemptions happen frequently due to the dense interferences. On the other hand, given the multiprocessor with multiple cores/caches, the cache access of tasks would be isolated with fewer preemptions, when tasks are executed on different cores with partitioning cache.

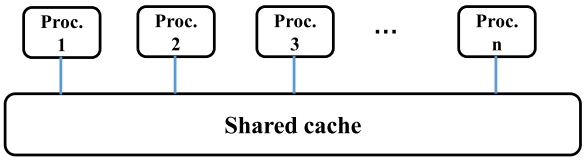


Figure 1: Cache model of the system: all cores share a same on-chip cache. Note that tasks executing on different cores do not have resource interference with each other although they share a common cache.

- C_i is the worst-case execution time for each job of the task.
- D_i is relative deadline for each job.
- T_i means each job of a task would released at least every T_i time units.

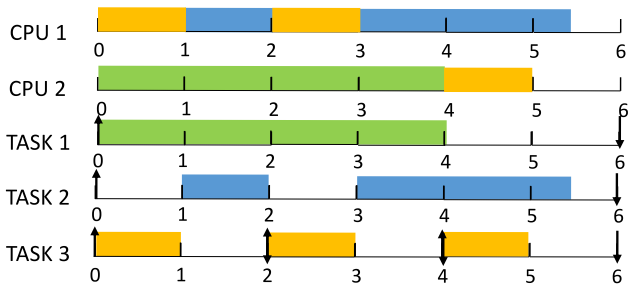
Each job has a absolute deadline d_i which occurs D_i time units after its release time.

We consider a constrained deadline in our system that $D_i \leq T_i$ holds for all tasks. We consider preemptive execution model, where during execution of a task, the executing job could be preempted or suspended at any instant of time, its execution may resume later on the same processor or another one.

Correctness. For a given scheduling algorithm, if all tasks can be scheduled without missing deadline based on the specification of the system, we defined the task set as schedulable.

Task	Period	Deadline	WCET
τ_1	6	6	4
τ_2	6	6	3.5
τ_3	2	2	1

(a) A sample task set with three tasks.



(b) The GEDF schedule of the task set show in Table 1(a) (with two processors), where all tasks are released at time 0. The second job of task τ_3 preempts task τ_2 at time instant $t = 2$ when all processors are busy, while the third job of task τ_3 is scheduled into an idle slot at time $t = 4$.

Figure 2: GEDF scheduling of a sample task set.

In this paper, we consider the GEDF scheduling algorithm

in the multi-processor system. GEDF is a dynamic scheduling algorithm which will place processes in a priority queue. The task's priority is assigned by the system based on their deadline. The task which has the earliest absolute deadline will have the highest priority; the task which has the latest absolute deadline will have the lowest priority [5].

Example II.1. *Consider the task set shown in figure 2(a), which can be correctly scheduled under GEDF (with the absolute deadline), as demonstrated in figure 2(b). Assuming a job arrived with an earlier absolute deadline, it is first scheduled into the idle time slots. If all the m processors are busy at that time instant, this newly released job would preempt the job with the lowest priority. Upon the completion of one processor, the processor would choose the pending jobs with the highest priority to execute.*

Observation 1. *Under the system and cache model shown in figure 1, for any newly released job to begin its execution at time t_0, if all processors are busy, the following two conditions must hold:*

- *Among the executing jobs, there are lower priority ones (i.e., with later absolute deadlines) than the job of interest.*
- *Only the job with lowest priority that was executing will be preempted while all other jobs will not be preempted – they will continue their executions until either there is a new job release or they are finished.*

Notation. Assume that a job with earliest absolute deadline has a higher priority. Let $hp(i)$ denote the set of tasks with smaller relative deadlines (and thus can preempt task τ_i); i.e.,

$$hp(i) = \{\forall \tau_j | D_j < D_i\}. \tag{1}$$

Let $P_j(D_i)$ denote the maximum number of jobs belonging to task τ_j that are invoked during a single job of task τ_i's execution period:

$$P_j(D_i) = max\Big(0, \lceil \frac{D_i - D_j}{T_j} \rceil\Big). \tag{2}$$

Let $E_i(t)$ represents that the maximum number of jobs, which have their release times and deadlines within the time interval of length t, of task τ_i can be invoked. We calculate it as follow:

$$E_i(t) = max\Big(0, 1 + \lceil \frac{t - D_i}{T_i} \rceil\Big) \tag{3}$$

Demand Bound Function. We use the Demand Bound Function $DBF(\tau_i, t)$ [12] to generate the maximum execution requirement by the jobs of τ_i that have both the arrival time and deadline within the time interval of length t. It can be calculated as follow:

$$DBF(\tau_i, t) = max\Big(0, \big(\lfloor \frac{t - D_i}{T_i} \rfloor + 1\big) \cdot C_i\Big). \tag{4}$$

Where C_i is the Worst Case Execution Time (WCET) for a task τ_i. In GEDF scheduling, tasks can execution in different cores simultaneously, Note that the inter-core interference when tasks are running is taken into account in WCET.

To analyze the CRPD, we use the concept of Useful Cache Block (UCB) and Evicting Cache Block (ECB).

Lee *et al*. [15] provided the definition for $UCBs$ as "A memory block m is called a useful cache block (UCB) at program point P, if it is cached at P and will be reused at program point Q that may be reached from P without the eviction of m". The memory blocks are loaded into the cache when a preempting task evicts other tasks, which are called $ECBs$. Combining the concept of UCBs and ECBs can assist us to bound CRPD.

III. GEDF-CRPD: A Multiprocessor GEDF Schedulability Test for CRPD Analysis

This section describes how CRPD analysis can be integrated into the existing schedulability test for GEDF on a multiprocessor platform. In order to do so, in Section III-A, we briefly introduce the widely accepted GEDF schedulability analysis without incorporating CRD. Then in Section III-B, we propose four different methods to integrate CRPD into demand bound functions in GEDF schedulability analysis.

A. Global EDF Schedulability Test

FLiu and Layland [18] explored the global multiprocessor scheduling of implicit deadline task. They gave a sufficient condition for guaranteeing that any tasks would not miss its the deadline.

$$u_{sum}(\tau) \leq m - (m-1) \cdot u_{max}(\tau). \tag{5}$$

In Equation (5), m denotes the number of processors, $u_{sum}(\tau)$ represents the total utilization and $u_{max}(\tau)$ represents the maximum utilization.

Later in 2007, Baker [3] [4] designed the GEDF schedulability test in an different perspective. He assumed that the task τ_k missed its deadline, then determined the necessary conditions for other tasks, that resulted in task τ_k to have missed its deadline. Finally, the negation of the necessary condition would have been a sufficient condition to guarantee all deadline being met for the task set.

In 2007, Baruah [5] designed a more sophisticated GEDF schedulability test that overcame some shortcomings of Baker's test. Similarly, he obtained a necessary condition which would let a job of task τ_k be the first to miss its deadline. When the necessary condition was not satisfied, then task τ_k would not have missed its deadline.

Based on this idea, if t_d is the time instance that a job of τ_k first missed its deadline, we use t_a to denote this job's arrival time, where $t_a = t_d - D_k$. if t_0 instant as the latest time instant $\leq t_a$, at which at least one processor is idle in

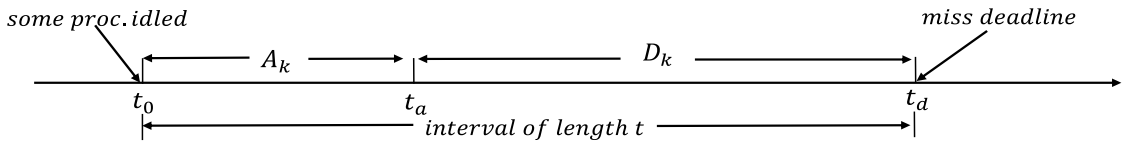


Figure 3: A job of task τ_k with arrival time at t_a and misses its deadline at t_d. t_0 is the time instant that at least one of the m processor is idle.

GEDF scheduling (Figure 3), In order to satisfy the deadline miss occurrence, it is necessary that all m processors are executing jobs other than τ_k's job more than $(D_k - C_k)$ time units in the time interval $[t_a, t_d]$. Hence, the total amount of execution requirement that execution in this interval t should satisfy:

$$\sum_{\tau_i \in \tau} I(\tau_i) > m \cdot (A_k + D_k - C_k). \quad (6)$$

We defined a time period $A_k = t_a - t_0$ in Equation (5), and $I(\tau_i)$ denotes the contribution of τ_i to work done in GEDF schedule during $[t_0, t_d]$.

If a task τ_i contributes no carry-in work[2] and the task τ_k does not miss its deadline, combined with the demand bound function, the contribution of τ_i to the total workload should not exceed the Equation (6).

$$I_1(\tau_i) = min(DBF(T_i, A_k + D_k), A_k + D_k - C_k). \quad (7)$$

Based on Baruah's work, we establish that the total amount of execution demand for tasks should not exceed the total amount of slack time period in m processors. The Equation (5) can be extended to the following format:

$$\sum_{i=1}^{n} max(0, \lfloor (t - D_i)/T_i \rfloor + 1) C_i \leq m \cdot (A_k + D_k - C_k). \quad (8)$$

Without considering the CRPD in GEDF scheduling, when the demand bound function satisfies Equation (6), the deadlines will be met. However, for GEDF in multiprocessor system, when a higher priority task preempts lower priority tasks, the introduced CRPD would enlarge the demand bound function significantly, the current sufficient condition cannot necessarily guarantee the schedulability of any sequence of tasks under GEDF scheduling. Figure 4 demonstrates that the given task set is no longer schedulable under GEDF with considering CRPD. Therefore, in the following subsection, CRPD will be integrated into the GEDF schedulability test framework introduced in this subsection.

B. Integrate CRPD into GEDF Scheduling

For a given task, DBF calculates the execution requirement in the interval of length t. When considering the CRPD into GEDF scheduling, the execution requirement for each job of the task τ_i should integrate the cache reload time γ_i.

$$\sum DBF(\tau_i, t) = \sum_{i=1}^{n} max\Big(0, \lfloor \frac{t - D_i}{T_i} \rfloor + 1\Big) \cdot (C_i + \gamma_i). \quad (9)$$

In the remainder of the subsection, we will present four different approaches in calculating and bounding the CRPD, γ_i.

(A) Ju's Approach. Ju *et al.* [13] presented an approach to integrate the CRPD analysis into uniprocessor EDF schedulability analysis in 2007. This approach first calculated the number of blocks belonging to τ_i that are directly preempted by task τ_j multiplied by $P_j(D_i)$. $P_j(D_i)$ is the maximum times that the task τ_j preempts a single job of task τ_i. In order to find all possible direct preemptions, the higher priority tasks which could preempt task τ_i are summed. These higher priority tasks τ_j represented as $j \in hp(i)$ and the γ_i^{Ju} represent CRPD calculated by Ju *et al.* [13]

$$\gamma_i^{Ju} = BRT \cdot \Big(\sum_{j \in hp(i)} P_j(D_i) \times \Big| UCB_i \cap ECB_j \Big| \Big). \quad (10)$$

where BRT is the per cache block reloading time. We modify Equation (8) and substitute γ_i which the value from Equation (9), so that the CRPD can be calculated in DBF for GEDF schedulability analysis.

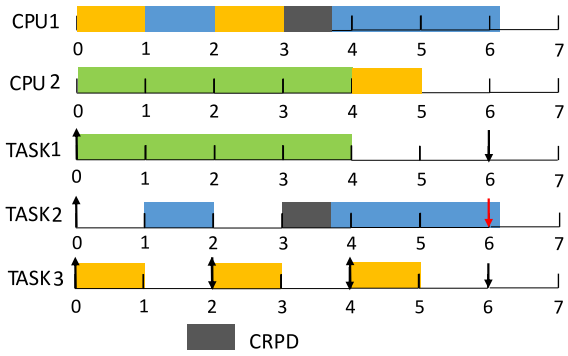


Figure 4: GEDF schedule of the taskset shown in Fig 2(a), where CRPD is taken into consideration and the first job of τ_2 misses its deadline at time $t = 6$.

[2]Carry-in work means that a job is released before t_0 and completes execution before t_d.

However, applying this method into GEDF would over-estimate CRPD as in uniprocessor. For instance, if a task τ_j could preempt τ_i in a time instant, but the task τ_i has already been preempted by a higher priority task τ_k, this approach will calculate all possible preemptions into τ_i's response time, which is overly pessimistically estimate the preemption times.

Lunniss *et al*. [19] provided an improved CRPD analysis for EDF scheduling in uniprocessor system in 2013. They used $\gamma_{t,j}$ to represent the $E_j(t)$ times of the preemptions cost for preempting tasks. Where $E_j(t)$ described the maximum number of jobs that belong to task τ_j. These jobs would have had their release times and absolute deadlines in an interval of length t.

We can apply this concept in CRPD analysis under GEDF scheduling. Therefore, the DBF could be changed into the following format:

$$\sum DBF(\tau_j,t) = \sum_{j=1}^{n} \Big(max\{0, \lfloor \frac{t-D_j}{T_j} \rfloor + 1\} \cdot C_j + \gamma_{t,j} \Big). \tag{11}$$

There are mainly two approaches for calculating $\gamma_{t,j}$ in Equation (11): (B) ECB-Union multiset[3], (C) UCB-Union multiset approaches and (D) Combined multiset approach, which are extended by Lunniss *et al* [19] to EDF scheduling based on the work of Staschlat *et al*. [22] in fixed priority for the uniprocessor.

(B) ECB-Union Multiset Approach. Nested preemptions make the pessimistic assumption, for any preemption by task τ_j, task τ_j itself may have already been preempted by a higher priority task. And total number of times that the jobs of task τ_k can be preempted by jobs of task τ_j is equal to $P_j(D_k) \times E_k(t)$. Therefore, the multiset $M_{t,j}$ could be formed as follow:

$$M_{t,j} = \bigcup_{\forall k \in \mathit{aff}(t,j)} \Big(\bigcup_{P_j(D_k) \times E_k(t)} \Big| UCB_k \cap (\bigcup_{h \in hp(j) \cup j} ECB_h) \Big| \Big) \tag{12}$$

In the time interval t for each processor.The job of task τ_j could at most invoke $E_j(t)$ times, therefore, ECB-union multiset approach bound the CRPD by summing the $E_j(t)$ largest value in the multiset $M_{t,j}$:

$$\gamma_{t,j}^{ecb-m} = BRT \cdot \sum_{l=1}^{E_j(t)} |M_{t,j}^{l}|. \tag{13}$$

where BRT is per block reloading time, $\gamma_{t,j}^{ecb-m}$ represents the CRPD calculated by ECB-Union multiset approach.

(C) UCB-Union Multiset Approach. This approach also use the concept of multiset. Lunniss *et al* [19] first form the multiset $M_{t,j}^{ucb}$. This multiset includes $P_j(D_k) \times E_k(t)$ times preemption of each task τ_k caused by task τ_j. Each time of preemption is represented by a set of cache blocks that might be preempted by task τ_j. Task τ_k whose relative deadline is greater that task τ_j's in the time interval [0, t) presented as $\mathit{aff}(t,j)$:

$$M_{t,j}^{ucb} = \bigcup_{\forall k \in \mathit{aff}(t,j)} \Big(\bigcup_{P_j(D_k) \times E_k(t)} UCB_k \Big). \tag{14}$$

Then they form the ECB multiset $M_{t,j}^{ecb}$, which contains the cache blocks that could be evicted by the jobs of task τ_j. Since τ_j invoked at most $E_j(t)$ times, the $M_{t,j}^{ecb}$ contains $E_j(t)$ times Repeated ECBs preempted by the a single jon of τ_j.

$$M_{t,j}^{ecb} = \bigcup_{E_j(t)} (ECB_j). \tag{15}$$

Finally, the intersection of $M_{t,j}^{ucb}$ and $M_{t,j}^{ecb}$ is multiplied by the BRT, they obtained the CPRD which is represent by $\gamma_{t,j}^{ucb-m}$:

$$\gamma_{t,j}^{ucb-m} = BRT \cdot |M_{t,j}^{ucb} \cap M_{t,j}^{ecb}|. \tag{16}$$

where BRT is per block reloading time, $\gamma_{t,j}^{ucb-m}$ indicates the CRPD calculated by UCB-Union multiset approach.

(D) Combined Multiset Approach. Since UCB-Union multiset and ECB-Union multiset approaches are not comparable [19], we get the minimum of these two result applying to the total DBF equation, which is represented as follow.

$$\sum_j DBF(\tau_j,t) = \sum_j min\{DBF(\tau_j,t)^{ucb-m}, DBF(\tau_j,t)^{ecb-m}\}. \tag{17}$$

where $DBF(\tau_j,t)^{ucb-m}$ indicates DBF obtained through UCB-Union multiset approach, Similarly, $DBF(\tau_j,t)^{ecb-m}$ represents DBF obtained by applying ECB-Union multiset approach.

Until now, we studied GEDF schedulability test on multiprocessor system and integrated the CRPD into GEDF in multiprocessor system. Moreover, we proposed four CRPDanalysis approaches under GEDF. However, ECB-union multiset approach, UCB-union multiset approach and combined multiset approach assume that each released job of tasks can causes a preemption of shared cache. The maximum number of preemption times is decreased in multiprocessor compared with uniprocessor. We will present improved CRPD analysis in Section IV.

IV. An Improved CRPD Upper Bound Analysis

In multiprocessor system, approaches (A), (B), (C) and (D) given in Section III-B usually over-estimate the CRPD under GEDF scheduling. Since they assume that each released job of tasks could generate a preemption cost. However, the cache interference of tasks would be reduced in

[3]Multiset is like a set, but it allows duplicate elements. For instance, {a, a, b} and {a,b} is not the same multiset. However, order does not matter. For example, {a, a, b} and {a, b, a} are the same multiset.

multiprocessor. Thus, we leverage the nature of the sparse interference between cache blocks distributed on multiprocessor, obtain a tighter bound of CRPD.

A. Condensing the Multiset

One of the main difference between uniprocessor and multiprocessor is that the first m tasks with the earliest relative deadline would not be preempted by other tasks in multiprocessor. According to observation 1, if one of these tasks begin to execute when released, it means that there exist some tasks in execution with an absolute deadline later than the first m task's absolute deadline. If the released task has a lower priority compared with some tasks with a latest absolute deadline, the task would wait until one of the jobs complete execution in m processors.

Multiset approaches would included all the useful cache blocks which may be evicted in the time interval of length t. Since the first m tasks with the earliest relative deadline would not be preempted, These approaches are all overestimate the affected cache blocks.

In ECB-Union multiset approach at Equation (12), when the task τ_k belongs to the taskset $\{\tau_1, \tau_2, \cdots, \tau_m\}$, the intersections between UCBs and ECBs are considered empty in multiprocessor system. Therefore, unless these values are not the l^{th} largest value in multiset M, the result would overestimate the CRPD. The equation below rectifies the limitation of Equation (12).

$$\left| UCB_k \cap \left(\bigcup_{h \in hp(j) \cup j} ECB_h \right) \right| = \emptyset, \quad k = 1, \cdots, m. \quad (18)$$

Similarly, in UCB-Union multiset approach, since the first m tasks will not be preempted, we can simply treat the UCB of these tasks as empty for calculations, then we obtain the following bound.

$$\left(\bigcup_{P_j(D_k) \times E_k(t)} UCB_k \right) = \emptyset, \quad k = 1, \cdots, m. \quad (19)$$

B. Reducing the Maximum Number of Preemptions

In the m multiprocessor system with GEDF scheduling algorithm (Figure 3), when we find that t_0 is the idle point in at least one processor, it means m jobs belonging to different tasks in execution in any time instant between t_0 and t_a. If a single job of task τ_i released at a time instant t_i, even if it has the earliest absolute deadline, it would only preempt the task τ_l with the latest absolute deadline. Other tasks are not interrupted by the task τ_l. In this situation, their response time would not be extended by preemptions respectively. The total number of invocation times for the higher priority tasks would be reduced compared with the times in uniprocessor. Therefore, the total preemption times can be decreased.

In multiprocessor, when a task is preempted, it could resume in any processor including the processor it utilized before. We also include this case into CRPD analysis since it could bring extra cache reload time. With this basic assumption, we mainly focus on how many preemptions occur in the time interval of length t.

In order to find the worst case preemption times, we first assume there are n tasks in the system and each single job of a task released would cause a preemption. We let the tasks with the latest m absolute deadline execute first, then the second m tasks with a higher priority release after. They could preempt all the tasks execution in the processor, in this sequence, until the task with the earliest absolute deadline released in one of the processor.

In this situation, the total preemption times should be $n - m$ when all jobs belong to different task completing the first time release. In fact, no matter what the sequence of tasks, the total preemption times in the first time invocation for different tasks, would not exceed $(n - m)$ times. Hence, we can subtract m^{th} least preemption cost from the total CRPD.

Through condensing the multiset M in combined multiset approach and refined the estimation of maximum preemption times. We further estimate the a tighter bound of CRPD in multiprocessor.

$$\sum_j DBF(\tau_j, t) = -\sum_{i=1}^{m} G^m + \sum_j min\{DBF(\tau_j, t)^{ucb-D}, DBF(\tau_j, t)^{ecb-D}\}. \quad (20)$$

where G denotes the interfered cache blocks for the first released job of each tasks. G^m is the m^{th} minimal interfered cache blocks set. We use $DBF(\tau_j, t)^{ucb-D}$ to represent the DBF calculated using condensed UCB-Union multiset approach and $DBF(\tau_j, t)^{ecb-D}$ to indicate the DBF calculated by condensed ECB-union multiset approach.

V. Experimental Evaluation

In this section, we evaluate the effectiveness of the different approaches to preemption cost computation on a large number of tasksets with varying taskset parameters. The task parameters used in our experiments were randomly generated as follows:

– The number of cores(m) are 2, 4, 8.

– The default task size is 15.

– The total number of task sets are 100.

– Task utilizations were generated using the UUnifast-discard algorithm [7].

– Task period were generated according to a uniform distribution with a factor of 100 difference between the minimum and maximum possible task period and a minimum periods of 5ms to 500ms, as found in most automotive and aerospace hard real-time applications.

– Task execution times were set based on the utilization and period selected: $C_i = U_i \cdot T_i$

– Task deadlines were implicit, i.e., $D_i = T_i$

– Priorities were assigned in deadline mon otonic order.

The following parameters affecting preemption costs are given below, the default values is given in parentheses:

– The number of cache-sets (CS=256).

– The cache reuse factor is 80%.

– The block-reload time (BRT = 8 μs)

– For each task, the UCBs of each task were assigned randomly based on [1].

The experiment shows how the integrated CRPD and global EDF schedulability analysis performed under the default configuration for implicit deadline taskset. We varied the utilization from 0.5 to m, and record how many tasksets were deemed schedulable by the global schedulability assuming no preemptions. Then we compared this experimental result with the cases under different CRPD analysis approaches in global EDF.

The Figure 5, Figure 6 and Figure 7 shows the result of 2 cores, 4 cores and 8 cores respectively. Each figure compared five approaches we proposed before. GEDF_CRPD_Ju describes the global EDF schedulability test for CRPD analysis based on Ju *et al*'s work [13]; i.e., Approach (A) in Section III-B. GEDF_CRPD_ECB represents the schedulability test based on ECB-union multiset approach; i.e., Approach (B) in Section III-B. GEDF_CRPD_UCB represents the schedulability test based on UCB-union multiset approach; i.e., Approach (C) in Section III-B. GEDF_CRPD_cb represents the schedulability test based on combined multiset approach (i.e., Approach (D) in Section III-B.) and GEDF_CRPD_cb represents the schedulability test based on condensed multiset approach with technology described in Section VI.

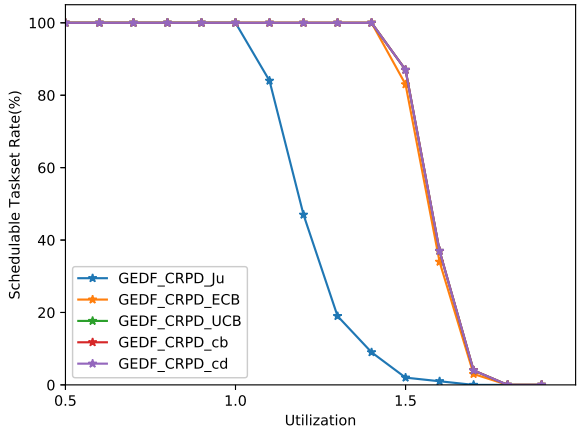


Figure 5: Evaluation for five CRPD analysis approaches: Number of tasksets could be schedulable at different total utilization in two processors.

After analyzing the figures, we find that GEDF_CRPD_Ju approach perform worst. Since it computes all possible preemptions caused by higher priority into a single job of tasks in DBF. Although the single direct preemption costs are precise, the total cost is very pessimistic. It overestimates the total cost of preemption. GEDF_CRPD_ECB and GEDF_CRPD_UCB approaches outperformed the Ju's approach. These two approaches have very close performance with our taskset. GEDF_CRPD_cb approach adopt the minimum value of GEDF_CRPD_ECB and GEDF_CRPD_UCB, therefore, It perform better than these two approaches sometime. Due to considering the sparse cache-block interference and refining the estimation of maximum preemption time in multiprocessor, GEDF_CRPD_cd approach has the best performance with our taskset. The experimental result shows that GEDF_CRPD_cd gives a tighter bound of CRPD.

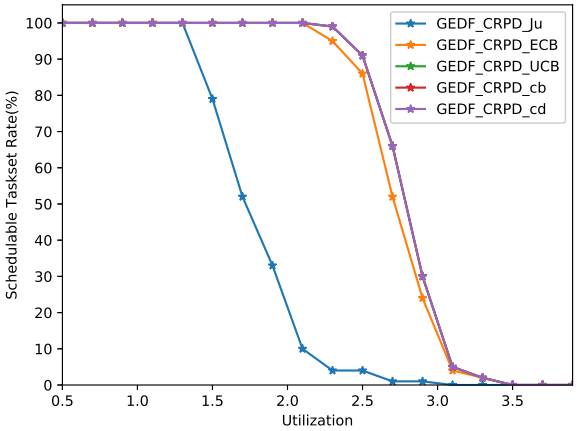


Figure 6: Evaluation for five CRPD analysis approaches: Number of tasksets could be schedulable at different total utilization in four processors.

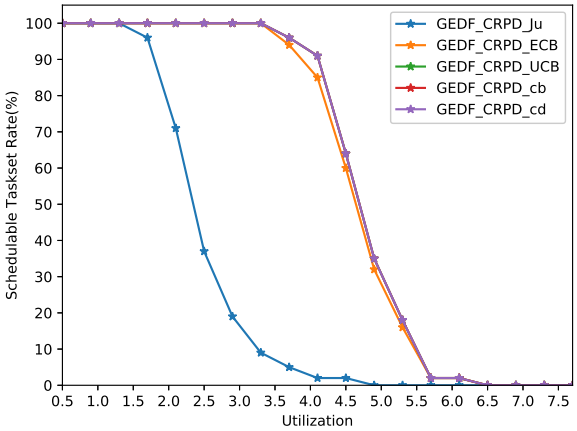


Figure 7: Evaluation for five CRPD analysis approaches: Number of tasksets could be schedulable at different total utilization in eight processors.

VI. Conclusion

In this paper, we first integrate the CRPD into GEDF schedulability test and present different methods to bound the CRPDunder GEDF. Specifically, condensed multiset approach leverages the ECB-union multiset approach and UCB-union multiset approach, so as to provide CRPD a tighter upper bound. Both theoretical analysis and the simulation results demonstrate the performance of the proposed method.

In the future, firstly, we aim to give a more precise method to calculate the total number of preemptions so that obtain a tighter bound of CRPD in schedulability analysis. Secondly, we aim at offering a more general approach which could be applied into different cache model to bound the CRPD.

Acknowledgment

Work supported by startup grants and a seed grant from Intelligent System Center at Missouri University of Science and Technology.

References

[1] S. Altmeyer, R. I. Davis, and C. Maiza. Cache related pre-emption delay aware response time analysis for fixed priority pre-emptive systems. In *IEEE 32nd Real-Time Systems Symposium (RTSS)*, pages 261–271. IEEE, 2011.

[2] S. Altmeyer, R. I. Davis, and C. Maiza. Improved cache related pre-emption delay aware response time analysis for fixed priority pre-emptive systems. *Real-Time Systems*, 48(5):499–526, 2012.

[3] T. P. Baker. Multiprocessor EDF and deadline monotonic schedulability analysis. In *IEEE 24th Real-Time Systems Symposium (RTSS)*, pages 120–129. IEEE, 2003.

[4] T. P. Baker. An analysis of EDF schedulability on a multiprocessor. *IEEE transactions on parallel and distributed systems*, 16(8):760–768, 2005.

[5] S. Baruah. Techniques for multiprocessor global schedulability analysis. In *IEEE 28th International Real-Time Systems Symposium (RTSS)*, pages 119–128. IEEE, 2007.

[6] M. Bertogna and M. Cirinei. Response-time analysis for globally scheduled symmetric multiprocessor platforms. In *IEEE 28th International Real-Time Systems Symposium (RTSS)*, pages 149–160. IEEE, 2007.

[7] E. Bini and G. C. Buttazzo. Measuring the performance of schedulability tests. *Real-Time Systems*, 30(1):129–154, 2005.

[8] P. Brucker. Multiprocessor tasks. In *Scheduling Algorithms*, pages 298–320. 1998.

[9] G. Buttazzo. *Hard real-time computing systems: predictable scheduling algorithms and applications*, volume 24. 2011.

[10] R. I. Davis and A. Burns. Resource sharing in hierarchical fixed priority pre-emptive systems. In *RIEEE International real-Time Systems Symposium (RTSS)*, pages 257–270. IEEE, 2006.

[11] R. I. Davis and A. Burns. A survey of hard real-time scheduling for multiprocessor systems. *ACM computing surveys (CSUR)*, 43(4):35, 2011.

[12] J. Goossens, S. Funk, and S. Baruah. Priority-driven scheduling of periodic task systems on multiprocessors. *Real-time systems*, 25(2):187–205, 2003.

[13] L. Ju, S. Chakraborty, and A. Roychoudhury. Accounting for cache-related preemption delay in dynamic priority schedulability analysis. In *Design, Automation & Test in Europe Conference & Exhibition*, pages 1–6. IEEE, 2007.

[14] H. Kim, D. De Niz, B. Andersson, M. Klein, O. Mutlu, and R. Rajkumar. Bounding memory interference delay in cots-based multi-core systems. In *IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 145–154. IEEE, 2014.

[15] C.-G. Lee, H. Hahn, Y.-M. Seo, S. L. Min, R. Ha, S. Hong, C. Y. Park, M. Lee, and C. S. Kim. Analysis of cache-related preemption delay in fixed-priority preemptive scheduling. *IEEE transactions on computers*, 47(6):700–713, 1998.

[16] C.-G. Lee, K. Lee, J. Hahn, Y.-M. Seo, S. L. Min, R. Ha, S. Hong, C. Y. Park, M. Lee, and C. S. Kim. Bounding cache-related preemption delay for real-time systems. *IEEE Transactions on software engineering*, 27(9):805–826, 2001.

[17] E. A. Lee. Cyber physical systems: Design challenges. In *IEEE 11th International Symposium on Object Oriented Real-Time Distributed Computing (ISORC)*, pages 363–369. IEEE, 2008.

[18] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM (JACM)*, 20(1):46–61, 1973.

[19] W. Lunniss, S. Altmeyer, C. Maiza, and R. I. Davis. Integrating cache related pre-emption delay analysis into edf scheduling. In *IEEE 19th Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 75–84. IEEE, 2013.

[20] H. S. Negi, T. Mitra, and A. Roychoudhury. Accurate estimation of cache-related preemption delay. In *Proceedings of the 1st IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis (ICHCSS)*, pages 201–206. ACM, 2003.

[21] F. Sebek. Measuring cache related pre-emption delay on a multiprocessor real-time system. *Memory*, 1024:66MHz, 2001.

[22] J. Staschulat, S. Schliecker, and R. Ernst. Scheduling analysis of real-time systems with precise modeling of cache related preemption delay. In *Euromicro 17th Conference on Real-Time Systems (ECRTS)*, pages 41–48. IEEE, 2005.

[23] Y. Sun and G. Lipari. Response time analysis with limited carry-in for global earliest deadline first scheduling. In *IEEE Real-Time Systems Symposium (RTSS)*, pages 130–140. IEEE, 2015.