多核处理器限制性可抢占 G-EDF 调度策略研究

韩美灵 邓庆绪 张天宇 冯智伟 林宇晗

(东北大学计算机科学与工程学院 沈阳 110819)

摘 要 多核处理器全局最早截止期优先(Global Earliest Deadline First,G-EDF)调度策略允许任务的抢占和任务在处理器之间迁移,频繁的抢占和核间迁移会导致较高的处理器开销,造成系统资源的浪费.然而目前针对多核处理器的可调度性分析方法都基于这样的假设:任务抢占和系统间迁移的开销计入最差响应时间或者忽略不计.但是实际研究表明该部分的开销在系统资源总开销中占重要部分,因此不可简单的忽略不计.而不可抢占调度,会给高优先级任务代入太多的阻塞从而导致其不可被调度.针对这类问题,实时领域的研究者们提出了限制性可抢占调度策略,且在全局固定优先级方面取得了很多的研究成果,然而在 G-EDF 方面的研究工作相对较少.该文研究了限制性可抢占全局最早截止期优先(Limited Preemption Global EDF,G-LP-EDF)调度策略,该策略结合了完全可抢占和完全不可抢占的优点.G-LP-EDF 调度策略把目前 G-EDF 最佳的分析方法和限制性可抢占调度策略相结合,目的是减少 G-EDF 的额外系统开销,避免系统资源的浪费,而不降低 G-EDF 的调度性.最后通过仿真实验,G-LP-EDF 分析方法在平均抢占次数上比 G-EDF 至少可减少 40%,而两个分析方法之间的可调性没有明显差距,大约为 1%.效率上两个方法随着最差执行时间的取值增大而增多,这是两个方法的本质造成的.然而 G-LP-EDF 整体比 G-EDF 的平均处理时间要慢,但差距都不足 1s.

关键词 多核处理器;实时嵌入式系统;限制性可抢占;最早截止期优先;偶发性任务集

中图法分类号 TP399 **DOI 号** 10.11897/SP.J.1016.2019.02355

Limited Preemption for G-EDF Scheduling on Multiprocessors

HAN Mei-Ling DENG Qing-Xu ZHANG Tian-Yu FENG Zhi-Wei LIN Yu-Han

(*Department of Computer Science and Engineering, Northeastern University, Shenyang* 110819)

Abstract With the increasing trend towards using multi-core architecture for embedded systems, the study of multiprocessor scheduling becomes attractive and desirable in the literature. G-EDF (Global Earliest Deadline First) is one of the most popular scheduling policy which allows preemption and migration between processors. There are many techniques proposed to derive an upper bound of the worst case response time or schedulable tests based on the assumption that the overheads of preemption and migration is omitted or just used an over-estimated value which is included in the worst case execution time. But there are works have been done to show that the overheads of preemption and migration cannot omit or include in the worst case execution time which would waste lots resource. And non-preemption does not have this problem but it would involve more blocks to the tasks with higher priority and make them un-schedulable. To combine the advantages of preemption and non-preemption, researchers in real-time area proposed the concept called limited-preemption. There are a lot of work about G-FP(Global Fixed-Priority) limited preemption, and the schedulability analysis of this problem is easier than G-EDF limited

收稿日期:2018-01-19;在线出版日期:2018-09-07.本课题得到国家自然科学基金(61472072,61528202)资助.**韩美灵**,博士研究生,中国计算机学会(CCF)会员,主要研究方向为嵌入式实时系统调度、多核嵌入式实时系统响应时间分析.E-mail:hanmeiling@stumail.neu.edu.cn.**邓庆绪**(通信作者),博士,教授,主要研究领域为物联网技术、嵌入式实时系统.E-mail:dengqx@mail.neu.edu.cn.**张天宇**,博士研究生,主要研究方向为嵌入式实时系统.**冯智伟**,博士研究生,主要研究方向为嵌入式实时系统.**林宇晗**,博士研究生,主要研究方向为嵌入式实时系统.

preemption. The method in G-FP limited preemption is not only reducing the preemption caused by higher priority tasks also improved the schedulability due to the character of G-FP scheduling method. The limited preemption has been well studied in G-FP scheduling while less has been done in G-EDF. Because the priority of each job of each task may be different in G-EDF while the priority of each job of each task is the same in G-FP. The differences involves much hardness to handle the preemption among each job of each task, and it is hard to find an available schedulability test of limited preemption G-EDF scheduling policy. Now, the G-EDF scheduling policy has been well studied on the multiprocessors which allowed preemption. But there no such a work to combine the state-of-the-art G-EDF scheduling method with the limited preemption. In this paper, we study the problem of limited-preemption scheduling of sporadic task systems which are running on multiprocessor platforms under G-EDF scheduling policy. The purpose of this paper is combine the state-of-the-art scheduling analysis method on multiprocessors under G-EDF with limited preemption. To avoid involving too much complexity, we just consider the total non-preemption interval among the scheduling of tasks for each task. To reduce the overheads induced by preemption and inter-processor migration under G-EDF scheduling, this proposed G-EDF limited preemption policy is denoted by G-LP-EDF, which avoids unnecessary preemption among tasks without losing the system schedulability but reducing the run-time cost. G-LP-EDF considers the scheduling of tasks and makes sure it cannot reduce the schedulability of tasks through combining the state-of-the-art scheduling method of G-EDF with limited preemption scheduling. In the end, we conduct sufficient experiments to show the precise and efficiency of this G-LP-EDF compared with the state-of-the-art method through simulation experiments. The performance of our proposed approach reduced tasks preemption at least 40% on average compared to the state-of-the-art method, and the schedulability difference between G-LP-EDF and the state-of-the-art method is the same. In efficiency, G-LP-EDF is slower than the state-of-the-art method and at most slower 20 ms which is not even 1 s.

Keywords multiprocessors; real-time embedded systems; limited preemption; earliest deadline first; sporadic task systems

1 引 言

随着计算机技术的迅猛发展,各种应用对处理器的性能需求也不断增加.由于能量和能耗的限制,典型的通过增加处理器速率提高处理器性能的方法遇到瓶颈.为了解决这一问题,多核处理器架构被提出.为了充分利用多核处理器的性能,学者们和应用开发者们尤其嵌入式实时领域的学者们对该领域的问题进行了大量的研究[1].多核处理器架构下的任务可调度性分析和单核比较,其关键是多核调度问题中任务的关键时刻的未知性.因此,多核处理器任务的可调度性分析是十分复杂和具有挑战性的问题.如何安排高优先级任务对低优先级任务的抢占减少由于任务抢占和迁移造成的系统开销,与此同时保证高优先级任务和低优先级任务的可调度性是诸多挑战中的一个.

抢占和迁移相关的开销包括:上下文切换的开销、缓存(cache)相关的抢占和迁移延迟、pipeline 延迟以及 bus 总线的开销[2-3].这些开销会增加任务在执行时的最差执行时间(Worst Case Execution Time,WCET),造成任务在可抢占调度策略下由于过多的抢占和迁移造成任务错失截止期.目前大部分的调度算法都基于悲观的 WCET 的假设即忽略抢占和迁移的开销,这么做的主要原因是:调度过程中存在大量可能会发生的抢占和抢占造成的迁移,这些抢占和迁移发生的时刻和次数是未知的;以及这些可能发生的抢占和迁移可能发生在调度开销比较大的地方,从而抢占发生时需要的缓存相关的开销会很大.因此,悲观的假设可以保证调度的正确性,却大大降低了系统资源的利用率.

完全不可抢占调度策略可以避免抢占和迁移造

成的开销，其代价是任务遭受大量的阻塞. 在多核处理器全局调度(Global Scheduling)中可抢占和不可抢占调度策略是不可比较的[4]. 例如：在多核全局调度中，任务集中有一个任务的执行时间很长(long WCET task)甚至超过一些任务的截止期，从而导致小截止期的任务不可被调度.

为了利用可抢占调度和不可抢占调度的优点，学者们提出一种限制性可抢占调度(Limited Preemption Scheduling，LPS)策略. LPS 策略可以减少由于抢占造成的开销，合理安排任务的执行提高任务的可调度性. LPS 的实现机制有多种，例如：在任务中设置可抢占点，不可抢占区域等[5]. 在多核处理器全局固定优先级调度(Global Fixed-Priority Scheduling，G-FP)中，固定可抢占点方法不仅可以减少抢占和迁移开销还可以正确的估计抢占和迁移的相关开销，且恰当的不可抢占点的设置可以提高任务的可调度性. 目前，固定可抢占点的设置问题已经取得了很多的研究成果[2,6-7]. 在 G-EDF(Global Earliest Deadline First)调度中，增加过多的不可抢占区域可能会降低任务的可调度性[8]，因此 G-EDF 和 LPS 结合的研究相对较少成为目前多核调度问题的一个研究热点.

G-EDF 在完全可抢占调度中已经取得了很多的研究成果. 文献[9]提出基于干涉量的响应时间分析方法. 文献[10]基于时间窗口分析改进了文献[9]的分析方法. 最近，文献[11]将两个文献结合，通过对时间窗口内干涉量的分析提出一种目前结果最好的 G-EDF 分析方法，下文简称该分析方法为 G-P-EDF.

本文针对 G-EDF 采用 LPS 的问题进行研究，发现目前该方面的研究相对较少. 近期，文献[12-13]针对多核 G-EDF 的 LPS 方法进行了分析，但是它们在进行可调度性分析时针对每个高优先级任务实例的干涉量的估算存在悲观性. 本文结合最新的 G-EDF 完全可抢占调度性分析的理论成果，提出一种 G-EDF 基于 LPS 的可调度性分析方法下文简称 G-LP-EDF. 主要思路是：首先假设每个任务的不可抢占区域的大小是已知的结合 G-P-EDF 分析方法分析任务的可调度性；根据任务的可调度性分析每个任务可以承受的最大抢占延迟. 本文的分析方法通过和 G-EDF 相结合，延迟抢占的发生提高任务的可调度性. 在多核中任务的可调度性和抢占次数是成负相关的，即不可抢占区域的长度越大任务遭受的阻塞量越大，任务的可调度性越低. 因此，本文引入一个抢占延迟函数，通过 G-P-EDF 分析最坏情况下每个任务实例在抢占发生时可以允许的最大延迟. 通过最大延迟函数和其遭受的最大阻塞，确定最终的最大阻塞量，从而保证不降低 G-P-EDF 的可调度性. 该分析方法的工作模式是工作保留模式(work-conserving)，即在某时刻如果存在任务实例没有完成执行那么该时刻所有处理器都处于忙碌状态.

我们将在第 2 节介绍 LPS 的相关工作和研究状态；在第 3 节介绍本文针对的系统模型；G-LP-EDF 的可调度性分析和抢占延迟函数的分析方法将在第 4 节进行介绍；最后在第 5 节通过仿真实验在可调度性、抢占次数以及效率三个层面对比 G-LP-EDF 和 G-EDF；在第 6 节总结本文工作.

2　相关工作

在实时系统任务的可调度性研究中，针对周期性任务和偶发性任务在多核完全可抢占调度策略下的可调度性分析已经取得了较多的研究成果[9-11,14]. 这些研究成果基于 WCET 的悲观估计忽略了抢占和迁移的开销，由于对抢占和迁移开销的悲观估计从而造成分析中任务错失截止期[15-16]. 文献[17]观察到跟缓存相关的抢占和迁移开销会造成任务的执行时间增加 33%，每一次抢占造成处理器高达 655 μs 的开销. 完全不可抢占调度策略可以避免抢占和迁移的开销，但是由于不可抢占从而导致任务被阻塞造成任务较低的可调度性而不受欢迎[18].

LPS 综合了完全可抢占和完全不可抢占的优点，成为学者们的研究热点. LPS 研究策略在单核的可调度性分析中取得了一定的研究成果[5-8]. 文献[19]提出单核固定优先级最优的 LPS 调度分析方法. 文献[20]基于文献[19]提出一种 G-FP 的延迟抢占调度策略，该方法通过合理的分配优先级和不可抢占区域长度提高了 G-FP 任务的可调度性，开启了 LPS 在多核 G-FP 领域的研究. 随后文献[21]纠正了文献[20]中的错误，并对文献[20]的理论成果进行了改善. 文献[18]把文献[21]的工作推广到固定可抢占点和多个不可抢占区域的系统模型中. 然而文献[18]只是针对 G-FP 的 LPS 分析方法提出了阻塞量的分析方法而没有提出全面的分析方法. 文献[22]改善了文献[18]工作的缺失，针对不同的抢占模式提出了任务的可调度性分析方法.

针对 EDF 的 LPS 分析方法相对较少. 在单核 EDF 调度中有文献[8,23-24],其中文献[8]第一次将 LPS 应用到 EDF 调度策略中,在不降低任务可调度性的前提下针对每一个任务计算其能承受的最大不可抢占区域. 文献[25]针对 EDF 在单核调度下的特性,提出一种可抢占和不可抢占的混合策略,不可抢占是针对整个任务而言,即任务集中的任务分成两种,一种执行在完全可抢占模式,一种执行在完全不可抢占模式. 该分析方法不能充分的利用 LPS 的特性. 针对 G-EDF 的 LPS 调度分析的研究相对较少,文献[12]第一次提出 G-EDF 的 LPS 调度分析方法,该方法通过触发抢占的方法进行分析,其基于文献[10]分析 G-EDF 的可调度性,存在悲观估计. 文献[13]提出一种基于G-EDF的最大阻塞量计算的LPS分析方法,并分析了该方法的最大加速比,是目前 LPS 在 G-EDF 策略上研究的最新理论成果,但其未能给出可调度性测试条件,实用性不强. 且文献[13]也是基于文献[10]的分析方法分析 G-EDF 中高优先任务实例的干涉量,然而文献[10]已经被文献[11]改善,该分析方法存在一定的悲观性.

3 系统模型

本文研究的实时嵌入式系统,由 M 个相互独立的同构处理器组成,处理器速率为单位时间. 在该系统中对任务集 $\tau=\{\tau_1,\tau_2,\cdots,\tau_n\}$ 进行调度. 任务集 τ 各个任务之间相互独立,且没有自挂起(self-suspension)发生. 在经典偶发任务模型中[2],每个偶发性任务由一个三元组表示:(C_i,D_i,T_i). 其中,C_i 定义为任务 τ_i 的 WCET;D_i 定义为任务的相对截止期(relative deadline),即任务的一个实例释放后必须在 D_i 时间单位内完成,否则该实例错失截止期;T_i 定义为任务 τ_i 两个连续释放的实例之间的最小时间间隔,通常称之为周期(period). 三个元素之间的关系是:$C_i\leq D_i\leq T_i$. 任务 τ_i 的利用率:$U_i=C_i/T_i$. 任务集的利用率 U_{tot} 定义为任务集中所有任务的利用率之和:$U_{\text{tot}}=\sum_{i=1}^{n}U_i$. 显而易见,一个任务集能被调度的必要条件为 $U_{\text{tot}}\leq M$.

在 LPS 中,每个任务增加一个不可抢占区域的参数,任务 τ_i 的不可抢占区域长度表示为 b_i,$0\leq b_i\leq C_i$. 所以任务 τ_i 的每个实例需要成 C_i-b_i 的可抢占区域的执行时间表示为 e_i 和不可抢占的执长度 b_i.

任务 τ_i 释放一个无限的任务实例序列 $J_i^1,J_i^2,\cdots,J_i^k,\cdots$. 假设任务 τ_i 的任意一个实例 J_i^k 在 r_i^k 时刻释放,在 f_i^k 时刻完成执行. 如果实例 J_i^k 满足截止期要求,则 $f_i^k\leq r_i^k+D_i$. 任务实例释放后,在完成之前的状态称为活跃(active)状态. 一个任务实例释放,表明当处理器空闲或者正在执行低优先级任务实例那么该实例就可以开始执行. 实例 J_i^k 的活跃区间为 $f_i^k-r_i^k$,此时间段又称之为实例 J_i^k 的响应时间 R_i^k. 任务 τ_i 的最差响应时间定义为所有任务实例响应时间的上界:$R_i=sup_k\{R_i^k\}$. 在本文中,讨论的问题是基于离散的时间模型,即所有的参数:释放时间、相对截止期和完成时间都是正整数.

本文采用 G-EDF 调度策略,即活跃的实例按照绝对截止期(absolute deadline)分配优先级,绝对截止期越早优先级越高. 任务 τ_i 在任意一个时间段 $t(t>0)$ 内释放的实例累积的最大执行时间之和可以根据需求函数(Demand Bound Function) $DBF(\tau_i,t)$[10]进行计算,具体如定义 1 所示.

定义 1. 任务 τ_i 在时间段 $t(t>0)$ 内释放的任务实例中释放时间和绝对截止期都在 t 内的实例所累积的最大执行时间需求定义为

$$DBF(\tau_i,t)=\max\left(0,\left(\frac{t-D_i}{T_i}+1\right)\cdot C_i\right) \quad (1)$$

在一个时间窗口 $[a,b)$ 内任务的实例可以按照其释放的时间、绝对截止期和时间窗口的关系分成三类(如图 1 所示):

(1) 前部任务实例(carry-in). 该实例的释放时间小于 a,截止期大于 a 小于 b,一个时间窗口只有一个该实例.

(2) 中部任务实例(body). 该实例的释放时间和截止期都大于 a 小于 b,且一个时间窗口内可以有多个该实例.

(3) 后部任务实例(carry-out). 该实例的释放时间大于 a 小于 b,但其截止期大于 b,且一个时间窗口只有一个该实例.

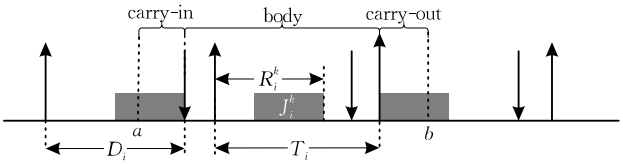


图 1 τ_i 时间 $[a,b)$ 窗口内的实例划分

4 G-LP-EDF 可调度性分析

本节的主要内容分成以下几个部分:首先,分析 G-LP-EDF 忙碌窗口内执行的实例的性质;其次,

根据忙碌窗口性质结合 G-P-EDF 分析方法提出 G-LP-EDF 分析方法，该方法假设每个任务的不可抢占区域是已知的；最后，针对每个任务计算其最大抢占延迟函数. 计算最大抢占延迟的基本思路是：根据 G-P-EDF 分析每个任务可以承受的抢占的最大延迟，从延迟 1 单位时间开始增加每个任务的响应时间，在每个延迟下使用 G-LP-EDF 分析任务的可调度性，然后更新最大延迟，直到任务的响应时间等于截止期或者任务错失截止期. 总而言之，为了保证该任务的可调度性，需要在新的忙碌窗口中采用 G-LP-EDF 进行分析，更新最大延迟，当抢占的延迟导致可以被 G-P-EDF 调度的任务变成不可调度时，该任务的抢占延迟为使任务错失截止期前的延迟.

本文研究 G-LP-EDF 的动机，可以通过例 1 进行说明.

例 1.　在一多核处理器嵌入式实时系统，设 $M=2$ 即该平台中有两个处理器. 设一个具有 $2n$ 个任务的任务集在该平台中运行. 设每个任务的 WCET 为 1，每个任务的相对截止期 $d_i=d_{2i}=i+1$，$1\leq i\leq n$. 每个任务的周期都为 $n+1$. 在每个时刻处理器选择具有最小截止期的任务在两个核上执行. 假设任务 τ_i 和 τ_{2i} 在 $(n-i)\varepsilon$ 处释放，$\varepsilon>0$ 且无限接近于 0. 在前 $n+1$ 时刻里发生 $4(n-1)$ 的抢占. 假如每次抢占延迟 ε 时刻即可避免抢占的发生，所有任务都在截止期内完成了执行.

如果任务的不可抢占区域的长度为 0，G-LP-EDF 可调度性分析和 G-P-EDF 一致. 即任务的可调度性分析变为完全可抢占调度. 在下文中除非特殊说明，任务都是按照 LPS 进行调度的.

4.1　G-LP-EDF 忙碌窗口的性质

任务的可调度性分析和 G-P-EDF 方法类似，我们针对每一个任务分析其可调度性. 在下文的论述中，任务 τ_k 为被分析任务. 针对任务 τ_k，G-LP-EDF 首先分析该任务可调度性判定的充分条件，然后针对每个任务用该充分条件去判定其可调度性从而保证整个任务集的可调度性.

假设任务集的一个合法实例序列采用 G-LP-EDF 调度策略进行调度时某个任务实例错失截止期. 假设任务 τ_k 的实例 J_k 是第一个错失截止期的任务实例. 该实例错失截止期的时刻表示为 t_d，设该错失截止的任务的释放时刻为 t_r. 显而易见，$t_r=t_d-D_k$.

定义 2.　时间窗口 $[t_0,t_d)$ 称为实例 J_k 的忙碌窗口，t_0 是满足以下条件的最晚时刻：

(1) $0\leq t_0<t_r$；

(2) 在 t_0 时刻之前至少有 1 个处理器完成了这样的实例：实例在 t_0 前释放，但绝对截止期不大于 t_d；

(3) 假如满足以上两个条件的 t_0 不存在，则 $t_0=t_r$.

定义 2 关于忙碌窗口的定义和经典的多核完全可抢占调度算法的定义相似. 不同的是在该问题的忙碌窗口内绝对截止期大于 t_d 的实例也可以阻塞 J_k 的执行，造成 J_k 错失截止期. 设 $A_k=t_r-t_0$，$L_k=t_d-t_0=A_k+D_k$，f_k 表示 J_k 可能的完成时间，$X_k=f_k-t_0$，如图 2 所示.

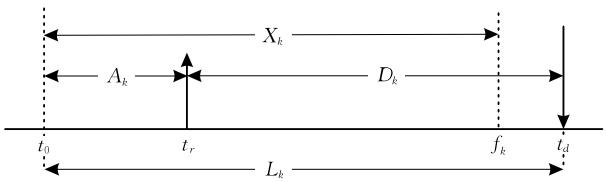


图 2　J_k 忙碌窗口示意图

根据 t_0 的定义，低优先级任务实例不可能在 $[t_0,t_r)$ 内开始执行，具体如引理 1 所示.

引理 1.　绝对截止期大于 t_d 的任务实例不会在 $[t_0,t_r)$ 内开始执行.

证明.　设实例 J_i 的绝对截止期大于 t_d，且在 $t_0+\varepsilon$ $(0\leq\varepsilon\leq A_k)$ 处开始执行，则在 $t_0+\varepsilon$ 时刻所有绝对截止期不大于 t_d 的任务实例完成了执行. 即在 $t_0+\varepsilon+\sigma$，σ 无限接近于 0，至少有一个处理器完成了所有在 $t_0+\varepsilon+\sigma$ 前释放的实例，且 $t_0+\varepsilon+\sigma<t_r$. 这一结论和 t_0 的定义不符.　　证毕.

根据定义 2，在 t_0 时刻最多有 $M-1$ 个高优先级任务实例在 t_0 之前开始执行，但是在 t_0 之后才完成执行，这样的任务实例称为前部任务实例(carry-in job). 那么在 t_0 时刻之前，至少有一个处理器完成了绝对截止期不大于 t_d 的任务实例. 根据 t_d 的定义，在 $t_0-\varepsilon\geq 0$ 时刻至少有一个低优先级任务开始了执行.

引理 2.　在 t 时刻，$t_r\leq t<t_d$，最多有 $M-1$ 个绝对截止期大于 t_d 的任务实例在 t 时刻同时执行.

证明.　由于 $t\in[t_r,t_d)$，t 时刻 J_k 未完成执行. 设 t 时刻 M 个任务实例进入不可抢占模式，即至少在 $t-\varepsilon$ 时刻，$t_r\leq t-\varepsilon<t_d$，有 M 个低优先级任务实例开始执行. 但是在 $t-\varepsilon$ 时刻实例 J_k 未完成执行的. 因此，在 $t-\varepsilon$ 时刻最多有 $M-1$ 个低优先级任务实例开始执行. 即 $[t_r,t_d)$ 内最多有 $M-1$ 个低优先级任务实例阻塞 J_k 的执行. 该引理得证.　　证毕.

任务实例的绝对截止期大于 t_d 的任务实例优先级低于 J_k，因此其在忙碌窗口内执行才会阻塞 J_k 的

执行.那么每个任务这样的实例只可能有1个,总结为结论1.

结论1. 在J_k的忙碌窗口$[t_0,t_d)$内,每个任务除了τ_k,只会有1个实例优先级低于J_k执行阻塞J_k的执行.

下面根据忙碌窗口的性质进行G-LP-EDF可调度性分析.

4.2 G-LP-EDF可调度性分析

任务τ_k的实例J_k在t_d处错失截止期,即在时间窗口$[t_r,t_d)$内J_k未完成C_k工作量的执行.那么J_k错失截止期的必要条件为在$[t_r,t_d)$内除了J_k以外的任务执行的工作量大于$M(D_k-C_k)$.根据G-P-EDF分析,在忙碌窗口内每个任务的执行可以分成两种类型:带carry-in的任务以及不带carry-in的任务.根据忙碌窗口的定义,任务集中最多有$M-1$任务带carry-in.因此,在忙碌窗口内的总干涉量可以定义为所有任务不带carry-in的情况加上一个补偿干涉.补偿干涉定义为最大的$M-1$带carry-in任务的干涉量和不带carry-in任务的干涉量的差值.该分析策略是一种经典的分析策略,详细解释可以参考文献[14].G-LP-EDF和G-P-EDF最大的不同是忙碌窗口中多了低优先级任务实例的阻塞量.因此,G-LP-EDF下任务的可调度性判定条件可以总结为定理1.

定理1. 任务集τ是G-LP-EDF可调度的,如果τ的每个任务τ_k满足以下判定条件:

$$\forall A_k\geqslant 0,\exists X_k\in[A_k,A_k+D_k]\,|\,X_k\geqslant C_k+\frac{1}{M}\Big(\sum_{\tau_i\in\tau}I_{i,k}^{NC}(X_k,A_k)+B_k(X_k,A_k)+\sigma_k\Big)\tag{2}$$

其中,$I_{i,k}^{NC}(X_k,A_k)$表示任务τ_i在长度为X_k的τ_k忙碌窗口内不带carry-in时产生的干涉量,σ_k表示任务τ_k在执行时受到的干涉量的补偿,即最大的$M-1$带carry-in和不带carry-in的干涉量差值之和.$B_k(X_k,A_k)$表示任务τ_k在忙碌期窗口内贡献的阻塞量.

证明. 在任务τ_k的忙碌期内,τ_k有可能在忙碌期开始后X_k长度内完成了执行,但其具体取值未知,因此需要对X_k的所有可能取值进行分析.在已知的X_k长度内任务的干涉量之和表示为当所有的任务不带carry-in时高优先级实例的不带carry-in的干涉量和补偿干涉量之和,其阻塞量由$B_k(X_k,A_k)$进行计算.任务τ_k只有在所有处理器都在执行这些干涉量时,才不能执行,因此定理1可以判定一个任务的可调度性.定理得证. 证毕.

在忙碌窗口$[t_0,t_d)$,τ_k不能执行的原因有两个:高优先级任务的干涉和低优先级任务的阻塞.

首先,高优先级任务实例的干涉分成两部分:不带carry-in的干涉量和补偿干涉量σ_k.高优先级任务实例的补偿干涉量定义为带carry-in的干涉和不带carry-in的干涉量的差值.

在忙碌窗口$[t_0,t_d)$内,任务τ_i释放的实例可以干涉τ_k执行且不带carry-in,那么最坏情况下该任务在$[t_0,t_d)$内执行的第一个实例在t_0处释放,且后续实例按照最小释放时间间隔释放(如图3所示),则其干涉量可由式(3)计算.

$$I_{i,k}^{NC}(X_k,A_k)=\begin{cases}\min(w_k^{NC}(X_k,L_k),I_k^{NC}(\max(0,L_k-D_k)),&i=k\\ \min(w_i^{NC}(X_k,L_k),X_k-C_k+1),&i\neq k\end{cases}\tag{3}$$

其中,$w_i^{NC}(X_k,L_k)$表示任务τ_i在窗口长度为X_k但是任务的绝对截止期不超过t_d的工作量上界,具体求解如式(4)所示.

$$w_i^{NC}(X_k,L_k)=N_i^{NC}\times C_i+\min(C_i,X_k-N_i^{NC}\times T_i)\tag{4}$$

其中,$N_i^{NC}=L_k-D_i/T_i$表示在窗口内释放的实例个数.当$i=k$时,τ_k有可能在窗口$[t_0,t_r)$内执行,其干涉量可以由式(5)计算:

$$I_k^{NC}(\max(0,L_k-T_k))=DBF(\tau_k,\max(0,L_k-T_k))\tag{5}$$

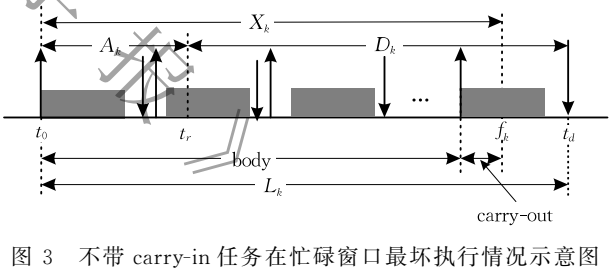


图3 不带carry-in任务在忙碌窗口最坏执行情况示意图

如果任务τ_i在忙碌窗口内执行的实例有两种:截止期大于t_d的实例和截止期不大于t_d的实例.根据结论1和引理2任务τ_i只可能有一个实例截止期大于t_d,且执行在$[t_r,t_d)$窗口内.当任务τ_i的某个实例J_i在忙碌窗口内以不可抢占模式执行,其提供的阻塞量定义为

$$B_{i,k}(L_k)=\min(C_i,b_i)\tag{6}$$

σ_k是所有带carry-in任务的干涉量和不带carry-in的干涉量的差值之中最大的$M-1$差值的和.根据引理1,低优先级任务实例不可能在窗口$[t_0,t_r)$内开始执行,且每个任务只可能有1个实例在该窗口执行在不可抢占模式,因此低优先级任务实例只有可能有1个作为带carry-in的任务执行在

窗口$[t_0,t_r)$内，该部分的阻塞量最大可表示为

$$B_k^{CI}(L_k)=\max\{B_{i,k}(L_k)\mid\tau_i\in\tau\backslash\tau_k\wedge D_i>L_k\}\quad(7)$$

下面分析任务 τ_i 带 carry-in 时的最差执行情况. 一个带 carry-in 的任务在忙碌窗口内能够执行的最大工作量基于以下假设：设任务的最后 1 个在窗口内执行的实例的绝对截止期等于 t_d，之前的实例按照最小释放时间间隔释放，如图 4 所示. 在窗口$[t_0,t_d)$内，任务 τ_i 的干涉量可以通过式(8)进行计算.

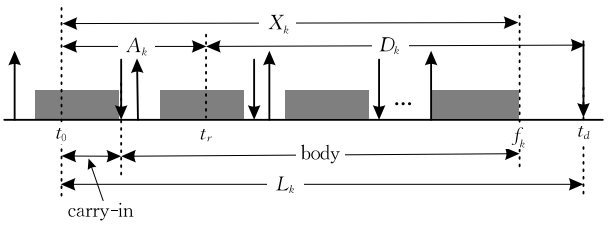


图 4　带 carry-in 任务在忙碌窗口最坏执行情况示意图

$$I_{i,k}^{CI}(X_k,A_k)=\begin{cases}\min(w_k^{CI}(X_k,L_k),I_k^{CI}(\max(0,L_k-T_k))), & i=k\\ \min(w_i^{CI}(X_k,L_k),X_k-C_k+1), & i\neq k\end{cases}\quad(8)$$

其中，$w_i^{CI}(X_k,L_k)$定义为任务 τ_i 带 carry-in 时在忙碌窗口内的工作量上限，可由式(9)计算：

$$w_i^{CI}(X_k,L_k)=\begin{cases}\min(X_k,\min(C_i,\max(0,L_k-(D_i-R_i)))), & p<0\\ \left(\dfrac{p}{T_i}+1\right)\times C_i+\min(C_i,\max(0,\alpha)), & p\geq 0\end{cases},$$

$$\alpha=p\bmod T_i-T_i+R_i\quad(9)$$

其中，$p=\min(X_k-C_i,L_k-D_i)$表示忙碌窗口内任务 τ_i 的高优先级实例的实际执行窗口. 当 $i=k$ 时，τ_k 有可能在窗口$[t_0,t_r)$内执行，其干涉量可以由式(10)计算：

$$I_k^{CI}(\max(0,L_k-T_k))=\frac{\max(0,(L_k-T_k))}{T_k}\cdot C_k+\beta,$$

$$\beta=\min(C_k,\max(0,(L_k-T_k)\bmod T_k-(D_k-R_k)))\quad(10)$$

每个任务 τ_i 的补偿干涉量定义为其带 carry-in 时的干涉和不带 carry-in 时的干涉的差值：

$$\sigma_{i,k}=I_{i,k}^{CI}(X_k,A_k)-I_{i,k}^{NC}(X_k,A_k).$$

则补偿函数 σ_k 可由式(11)进行计算：

$$\sigma_k=\sum_{\text{the largest }M-1}\sigma_{i,k}\quad(11)$$

假如定理 1 中对 A_k 的取值没有界限，那么该可调度性判定条件是不成立的，因为要枚举的时间窗口是无限多的. 结合任务集合窗口$[t_0,t_r)$的特性，可以得到 A_k 的取值上限，如引理 3 所示.

引理 3.　A_k 是任务 τ_k 忙碌期窗口的扩展窗口$[t_0,t_r)$的长度，则 $A_k\leq\widetilde{A}_k$：

$$\widetilde{A}_k=\frac{C_M+\sum_{\tau_i\in\tau}(T_i-C_i)\times U_i}{M-U_{\text{tot}}}\quad(12)$$

证明.　首先当一个任务不带 carry-in 时，在一个长度为 t 的时间窗口内可以产生的最大工作量之和为

$$w_i(t)=\frac{t}{T_i}\times C_i+\min(C_i,t\bmod T_i).$$

显而易见，$w_i(t)$严格小于如下线性方程：

$$lw_i(t)=U_i\times t+(T_i-C_i)\times U_i.$$

在窗口$[t_0,t_r)$内的执行的任务的最坏情况发生在：当所有的前部任务实例的工作量都为其最差执行时间，由引理 1 可知该窗口内没有低优先级任务实例开始执行，即使是低优先级 carry-in 实例最大也只可能贡献其 WCET 的阻塞量，因此所有 carry-in 实例的干涉量之和为任务集中 M 最大的 WCET 之和表示为 C_M. 那么$[t_0,t_r)$内执行的工作之和肯定不小于窗口可以提供的工作量，即：

$$C_M+\sum_{\tau_i\in\tau}lw_i(t)>M\times A_k.$$

代入 $lw_i(t)$，进行公式变换最终我们得到：

$$A_k<\frac{C_M+\sum_{\tau_i\in\tau}(T_i-C_i)\times U_i}{M-U_{\text{tot}}}.$$

该引理得证.　证毕.

至此为止，我们结合 G-P-EDF 分析方法给出了 G-LP-EDF 可调度性分析方法. 下面结合两种方法，我们分析每个函数的最大阻塞量.

4.3　最大抢占延迟函数

为了保证 G-P-EDF 的可调度性不受影响，我们引入最大抢占延迟函数，来决定每个任务抢占发生时可以被延迟的最长时间. 抢占被延迟则低优先级任务实例继续执行，造成低优先级任务实例对高优先级任务的阻塞. 由于低优先级任务的阻塞导致 X_k 变大. 根据定理 1，窗口长度变大则窗口内高优先级任务实例的干涉量可能会增大，因此需要在新的窗口中采用定理 1 重新分析任务的可调度性.

我们用 $B_k(X_k,A_k)$表示任务 τ_k 在 $t_r+X_k-A_k$ 处完成执行时，低优先级任务实例的阻塞量之和. 根据定理 1，引理 4 成立.

引理 4.　如果任务 τ_k 可以被调度，$\forall A_k$，$\exists X_k\in[A_k,A_k+D_k]$，以下不等式肯定成立：

$$B_k(X_k,A_k)\leq M\times(X_k-C_k)-\sigma_k-\sum_{\tau_i\in\tau}I_{i,k}^{NC}(X_k,A_k)\quad(13)$$

证明. 该引理显而易见是成立的,需要注意的是补偿干涉量 σ_k 中包含低优先级 carry-in 实例的阻塞量,其应该包含到所有低优先级任务的阻塞量中.假如任务 τ_k 可以满足截止期要求,那么其必须在忙碌期内完成 C_k 的工作量.根据定理 1,在忙碌期窗口内执行的任务实例分为三种,一种比 τ_k 的实例优先级高的实例其干涉量之和不大于 $\sigma_k+\sum_{\tau_i\in\tau}I_{i,k}^{NC}(X_k,A_k)$,低优先级任务的阻塞量 $B_k(X_k,A_k)$ 和 τ_k 实例本身的执行.该引理得证. 证毕.

X_k 的最大取值为 $X_k=A_k+D_k$.此时,任务 τ_k 获得的干涉量最大,干涉量和窗口长度有直接关系,τ_k 要满足截止期则必须在处完成执行,因此无论 X_k 在 $[A_k,A_k+D_k]$ 范围内如何取值其干涉量都不会大于当 $X_k=A_k+D_k$ 的干涉量.综上,得到以下定理.

定理 2. 任务 τ_k 允许的最大强抢占延迟为

$$Q_k\le\min_{A_k\in[0,\tilde{A}_k]}\left\{M\times(A_k+D_k-C_k)-\left(\sigma_k+\sum_{\tau_i\in\tau}I_{i,k}^{NC}(X_k,A_k)\right)\right\}\tag{14}$$

其中,每个任务的最大不可抢占区域长度都为 0,因此 σ_k 中不包含任何阻塞量.

证明. 当抢占发生时,最大的目的是在保证任务可调性的前提下进行抢占的延迟操作,每种情况下允许的最大延迟是不同的,需要在任何情况下都能保证任务的可调度性,因此取所有情况的最小值. 证毕.

根据任务的可调度性分析,我们很容易得到新的可调度性判定条件,即当任务允许最大延迟小于 0 时,说明在该任务的忙碌期内不能完成所有高优先级任务的执行,因此任务错失截止期,因此以下结论成立:

结论 2. 如果任务 τ_k 可以被调度,当且仅当其最大抢占延迟满足:$Q_k\ge 0$.

接下来,我们综合以上分析,分析如何根据最大抢占延迟分析任务每个抢占的最大延迟时间.在 G-LP-EDF 调度下,每个实例的执行最多有两种模式:可抢占模式(preemption model)和不可抢占模式(non-preemption model).设在 t_h 时刻,M 个处理器上执行的任务实例集合是 $\{J_{p_1},J_{p_2},\cdots,J_{p_M}\}$,每个任务实例剩余的执行时间是 $\{e_{p_1},e_{p_2},\cdots,e_{p_M}\}$,它们的绝对截止期分别是 $\{d_{p_1},d_{p_2},\cdots,d_{p_M}\}$.在 t_h 时刻,释放了实例 J_h,该实例的截止期是 d_h,且 $d_h<\max_{i=1}^{M}(d_{p_i})$.设 $P_{\text{set}}=\{J_1,J_2,\cdots,J_{p_n}\}$ 为正在执行且优先级低于 J_h 的任务实例集合.设实例 J_h 的最大抢占延迟函数为 Q_h.当处理器已经执行完 P_{set} 中的某个实例或者最大延迟函数已经到达,J_h 都必须开始执行,因此实际上 J_h 导致的抢占被延迟的最大时间为 P_{set} 实例中剩余的最小的执行需求和延迟函数的最小值,该结论可以总结为引理 5.

引理 5. J_h 的执行最多被延迟 S_h:

$$S_h=\min\left(\frac{Q_h}{|P_{\text{set}}|},\min\{e_{p_i}\mid J_{p_i}\in P_{\text{set}}\}\right)\tag{15}$$

证明. 设 J_h 在 t_h 时刻,且 J_h 的优先级高于 P_{set} 中所有任务实例的优先级.在 G-P-EDF 调度中 J_h 要抢占 P_{set} 中优先级最低的任务.在 G-LP-EDF 中,该抢占被延迟,根据 $\min(e_i\mid J_i\in P_{\text{set}})$ 和 $Q_h/|P_{\text{set}}|$ 的大小关系,可分两种情况讨论引理的正确性:

情况 1. $\min(e_i\mid J_i\in P_{\text{set}})>Q_h/|P_{\text{set}}|$.

假设 $S_h=\min(e_i\mid J_i\in P_{\text{set}})$,$J_h$ 不会错失截止期.此时 J_h 承受的低优先级任务量之和为

$$\min(e_i\mid J_i\in P_{\text{set}})\times|P_{\text{set}}|<Q_h.$$

该结论和情况 1 的条件相违背.

情况 2. $\min(e_i\mid J_i\in P_{\text{set}})\le Q_h/|P_{\text{set}}|$.

假设 $S_h=Q_h/|P_{\text{set}}|$,J_h 不会错失截止期.在 $\min(e_i\mid J_i\in P_{\text{set}})$ 时间之后,处理器会处于空闲直到 J_h 开始执行,这和 G-LP-EDF 是工作忙碌模式的假设不符.

该引理得证. 证毕.

根据定理 2 和引理 5 可以直接计算任务的抢占延迟,但是不能确定延迟是谁造成的,不能采用定理 1 分析任务的可调度性.因此需要结合定理 1,分析每个任务的不可抢占区域的长度.根据上述分析,为了分析假设任务的抢占延迟一次发生在 carry-in 窗口一次发生在 $M-1$ 个低优先级任务的并行执行.注意,如果一个任务的不可抢占区域过长,那么很容易导致其他任务错失截止期,目的是避免某些任务实例执行到快结束时其执行不会被中断,因此分散的较小的不可抢占区域长度是可取的.具体的每个任务的不可抢占区域的分析可以按照以下步骤完成:

(1)设任务集中每个任务的响应时间为其相对截止期,且存储在 RT 中,且任务的不可抢占区域为 0;

(2)根据引理 4,计算每个任务的最大允许的阻塞量;设被分析任务为 τ_k,在除 τ_k 外的任务中,选择具有最小阻塞量的任务增加该任务的不可抢占区域

增加 1；然后采用定理 1 判定该任务的可调度性，更新最大阻塞量和响应时间；

(3) 对任务 τ_k 重复步骤(2)，直到其响应时间等于截止期；

(4) 对每个任务重复步骤(2)和(3)，直到每个任务的松弛为 0 即没有空间让步给低优先级任务实例执行.

到目前为止，我们分析了 G-LP-EDF 调度策略的可调度性，且结合 G-LP-EDF 提出了最大抢占延迟的计算方法. 该过程重复进行了多次可调度性的判定，由于增加过多的不可抢占区域会导致任务错失截止期. 增加幅度过大任务就容易错失截止期，因此每个任务的不可抢占区域略小且分散可以保证当任务执行到剩余很小的执行时间时，避免抢占的发生. 虽然复杂度略大于 G-P-EDF，但是通过效率实验发现该分析方法并不会增加太多分析时间. 下面我们通过仿真实验来对比 G-LP-EDF 分析方法和 G-P-EDF 分析方法的可调度性、抢占次数的变化以及效率的差异.

5　仿真结果与分析

本节通过仿真实验，从可调性、抢占次数以及效率等三个方面对本文提出的 G-LP-EDF 分析方法和 G-P-EDF 方法进行对比.

5.1　仿真环境

本仿真实验基于 Windows 平台，采用 python 工具进行分析方法的仿真. 仿真实验主要分成两部分：任务集的随机生成方法和每个任务集中任务的可调度性判定条件的实现.

(1) 随机生成任务集的仿真

任务集中的任务个数 n 的取值为：$\{M+3, 2\times M, M+10\}$，M 为处理器个数本实验中处理器个数取值为 $\{2,4,8\}$. 首先根据任务集的利用率之和 $U_{\text{tot}} \in \{0.5, M\}$，采用均匀分布(Uniform Distribution)把利用率分配给 n 个任务. 每个任务的周期采用对数正态分布(logarithmic normal distribution)，取值范围[100,1000].

每个任务的 WCET 通过 $\max(1, u_i \times T_i)$ 计算. 每个任务的相对截止期采用均匀分布在 $[\max(C_i, T_i/2)]$. 设在 0 时刻所有任务都释放了任务实例，之后每个任务按照最小释放间隔释放.

(2) 判定条件的仿真

针对每一个任务集，判定其中任务的可调度性. 针对每一个任务，根据初始响应时间集合 RT 和不可抢占区域长度 0，采用定理 1 判定其是否满足定理 1 的判定条件，假如任务集中的所有任务都满足则任务集可以被调度.

G-P-EDF 采用定理 1 的判定条件，前提是任务的不可抢占区域的长度为 0；G-LP-EDF 采用定理 1 判定其可调度性，每个任务具有最大的抢占延迟.

5.2　仿真结果

(1) 可调度性实验

本节设处理器个数设为：$M=8$，每个任务的利用率最大为 0.5(多个利用率小的任务较少的利用大的任务容易被调度). 采用接受率表示任务的可调度性，接受率定义为可被调度的任务集个数占所有被分析的任务集个数的比率.

如图 5 所示，横坐标表示任务集的规范化利用率 U_{nor}($U_{\text{nor}}=U_{\text{tot}}/M$)，纵坐标表示在某个利用率范围内的任务的接受率. G-LP-EDF 和 G-P-EDF 的接受率曲线完全重合，说明了 G-LP-EDF 并未降低 G-P-EDF 的调度性.

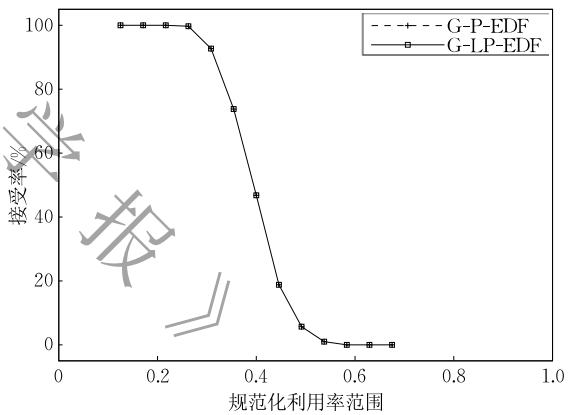


图 5　$M=8$，可调度性对比实验

(2) 抢占次数实验

不同任务实例序列可能会导致更多的抢占发生，因此我们通过统计足够长时间内的平均抢占次数和最大抢占次数进行对比.

为了验证本文提出的 G-LP-EDF 是否达到减少抢占次数而不降低 G-P-EDF 调度性的目的，通过随机生成任务集来进行模拟调度实验. 通过在一个足够大的时间区间调度任务集，统计调度中的任务的抢占次数比较 G-P-EDF 和 G-LP-EDF 平均抢占次数的差距和分析样本中最大抢占次数之间的差距分析 G-LP-EDF 在减少抢占次数上的优势.

统计每个任务集 τ^k 在 $[0, 10^6]$ 内产生的抢占次

数是 φ^k，k 的最大取值是 1000. 每组变量(n,U_{tot})进行实验，对两组实验结果进行统计：

①1000 个任务集中，抢占次数的平均值表示为 N_{av}：

$$N_{av}=\frac{\sum_{k=0}^{1000}\varphi^k}{1000}.$$

②1000 任务集中最大的抢占次数表示为 N_{max}：

$$N_{max}=\max_{k=0}^{1000}\varphi^k.$$

关于平均抢占次数的实验结果如图 6～图 8 所示，最大抢占次数的实验结果如图 9～图 11 所示. 实验的处理器个数 $M=8$. 图 6～图 11 的横坐标是采用任务集的规范化利用率. 而图 6～图 8 的纵坐标表示为平均抢占次数，图 9～图 11 纵坐标表示为最大的抢占次数. 图 6～图 11 中 G-P-EDF 对应的曲线，表示该实验结果采用的是 G-P-EDF 调度算法，而 G-LP-EDF 对应的曲线表示采用的是 G-LP-EDF 调度算法.

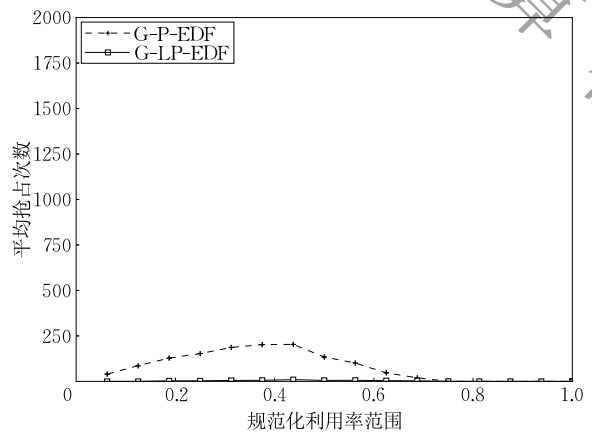


图 6 $n=m+3$ 平均抢占次数对比实验

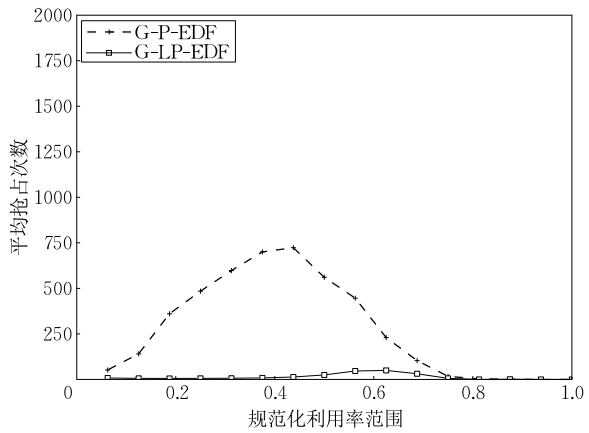


图 7 $n=2m$ 平均抢占次数对比实验

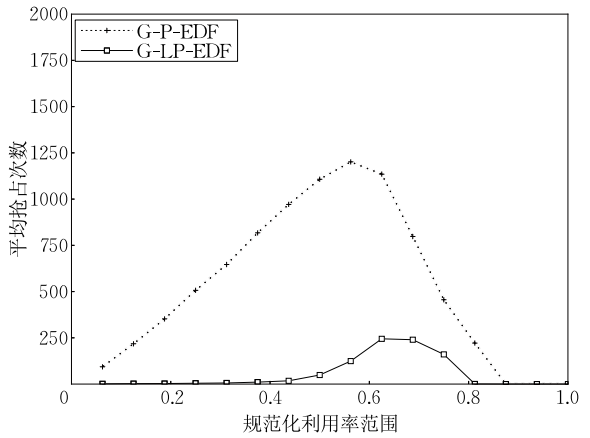


图 8 $n=M+10$ 平均抢占次数对比实验

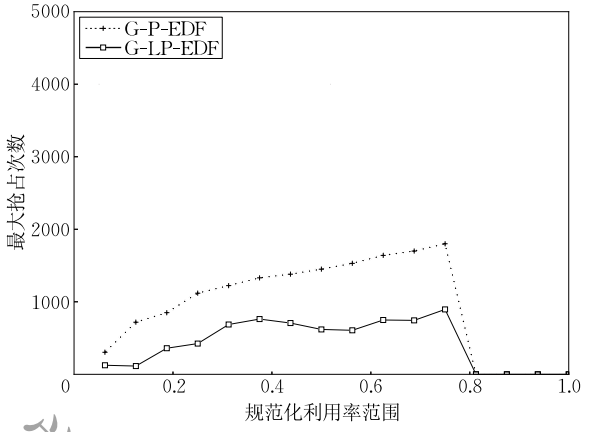


图 9 $n=m+3$ 最大抢占次数对比实验

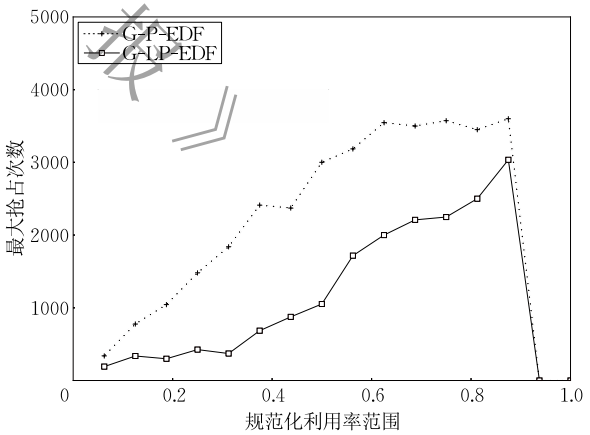


图 10 $n=2m$ 最大抢占次数对比实验

图 6～图 8 的实验结果表明，随着任务集中任务个数的增加平均抢占次数的最大值也呈上升趋势. 图 6～图 8 的趋势表明，随着任务集利用率的增加任务的可抢占次数首先升高，当利用率太大时由于可调度性逐渐降低因此抢占次数也减少. 而两个曲线的对比表明，G-LP-EDF 和 G-P-EDF 的可调度性是一致的. 图 6～图 8 中曲线的趋势也表明

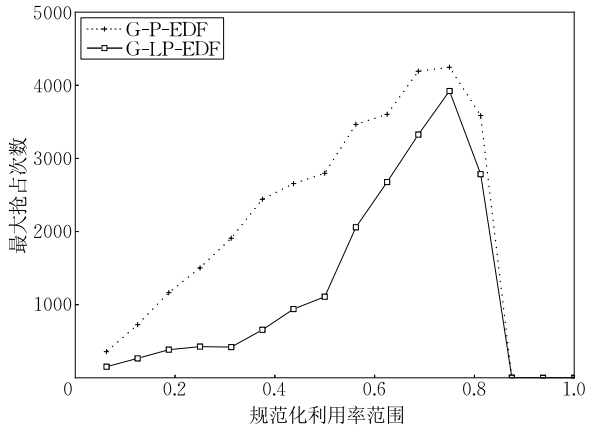


图 11　$n=m+10$ 最大抢占次数对比实验

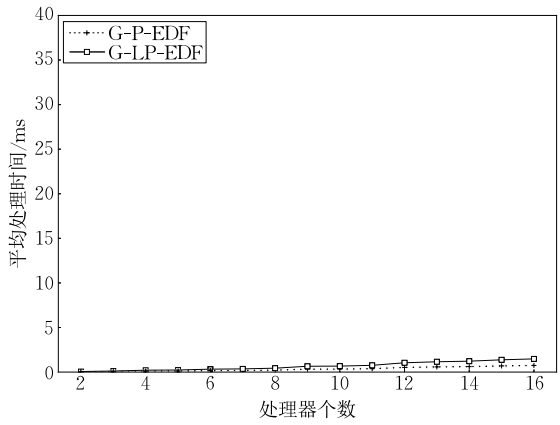


图 12　不同处理器个数的效率对比实验

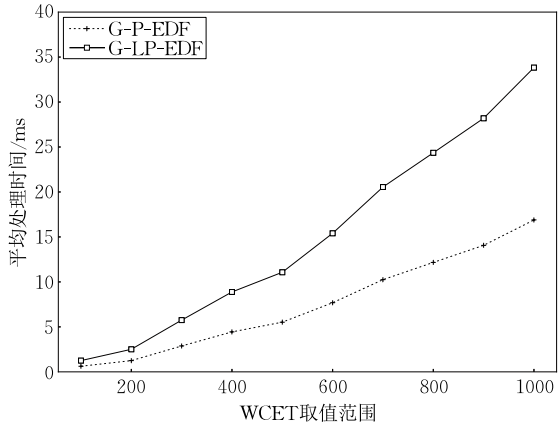


图 13　不同 WCET 取值范围的效率对比实验

当任务集的利用率比较小或者任务个数比较少时，G-LP-EDF 几乎可以避免全部的抢占. 即使在抢占高发的利用率区间，G-LP-EDF 也可以避免至少 40％的抢占，图 8 中两个波峰的差值比.

图 9～图 11 的实验结果表明，两个算法在最大抢占次数曲线之间对比的差距小于平均值的对比，说明针对某些任务集 G-LP-EDF 算法并不能明显的减少 G-P-EDF 算法的抢占次数. 同样随着任务集利用率的增加任务的可调度性减少，而曲线的走势表明两者的可调度性一致.

图 6～图 11 的实验对比，表明本文提出的 G-LP-EDF 算法明显减少了 G-P-EDF 的平均抢占次数，然而该结果并不是针对某些特定的任务集而是具有普遍的适用意义.

(3) 效率实验

本节对比了 G-LP-EDF 分析方法和 G-P-EDF 分析方法之间的效率问题. 第一个效率实验，通过不同的处理器个数，相同的 WCET 取值范围和任务集中任务个数的条件下，两个算法分别处理 1 个任务集的平均处理时间进行对比. 第一个效率实验如图 12 所示，处理器个数的取值范围为[2，10]步长为 1，WCET 取值范围为[1，500]，任务集中任务的个数为 15. 第二效率实验，在相同的处理器个数和任务集中任务的个数下，通过不同的 WCET 取值对两个分析方法分别处理 1 个任务集的平均时间进行对比. 该实验如图 13 所示，处理器个数为 8，任务集中任务的个数为 15，WCET 的取值范围为[1000，10000]步长为 1000.

如图 12、图 13 所示，虽然 G-LP-EDF 从两个效率实验上和 G-P-EDF 对比有所下降，但是完全在可接受范围内，即 G-LP-EDF 分析方法可以在较短时间内处理较复杂的任务集.

6　结　论

(1) 本文为了减少 G-P-EDF 调度算法的抢占次数，避免系统资源的浪费提出多核 G-LP-EDF 分析方法. 该方法的目的是在不降低 G-P-EDF 可调度性的前提下减少任务抢占和迁移导致的额外开销.

(2) 本文结合最近提出的 G-EDF 分析方法，提出了 G-LP-EDF 分析方法的可调度性判定条件，并对其正确性进行了证明.

(3) 最后通过随机仿真实验表明 G-LP-EDF 调度策略在足够长的时间内平均抢占次数上至少能减少 40％左右，当利用率和任务集中任务个数相对较少时甚至可以达到完全避免任务的抢占. 且其可调度性并没有降低. 但是效率上确实较 G-P-EDF 慢但是每个任务集的平均处理速度在毫秒级别因此是可以接受的.

(4) 未来的工作我们打算将该算法模型和并行任务模型相结合，在并行任务模型上实现 G-LP-EDF 的调度.

致 谢 本工作受到辽宁重大装备制造协同创新中心资助,在此表示衷心的感谢!

参 考 文 献

[1] Baruah S K, Mok A K, Rosier L E. Preemptively scheduling hard-real-time sporadic tasks on one processor//Proceedings of the 11th Real-Time Systems Symposium. Lake Buena Vista, USA, 1990: 182-190

[2] Bertogna M, Buttazzo G, Marinoni M, et al. Preemption points placement for sporadic task sets//Proceedings of the 22nd Real-Time Systems. Brussels, Belgium, 2010: 251-260

[3] Brandenburg B, Anderson J H. Scheduling and Locking in Multiprocessor Real-Time Operating Systems [Ph. D. dissertation]. University of North Carolina, USA, 2011

[4] Guan N, Yi W, Gu Z, et al. New schedulability test conditions for non-preemptive scheduling on multiprocessor platforms//Proceedings of the 29th IEEE Real-Time Systems Symposium. Barcelona, Spain, 2008: 137-146

[5] Buttazzo G C, Bertogna M, Yao G. Limited preemptive scheduling for real-time systems. A survey. IEEE Transactions on Industrial Informatics, 2013, 9(1): 3-15

[6] Bertogna M, Xhani O, Marinoni M, et al. Optimal selection of preemption points to minimize preemption overhead//Proceedings of the 23rd Euromicro Conference on Real-Time Systems. Porto, Portugal, 2011: 217-227

[7] Peng B, Fisher N, Bertogna M. Explicit preemption placement for real-time conditional code//Proceedings of the 26th Euromicro Conference on Real-Time Systems. Madrid, Spain, 2014: 177-188

[8] Baruah S. The limited-preemption uniprocessor scheduling of sporadic task systems//Proceedings of the 26th Real-Time Systems. Miami, USA, 2005: 137-144

[9] Bertogna M, Cirinei M. Response-time analysis for globally scheduled symmetric multiprocessor platforms//Proceedings of the 28th Real-Time Systems Symposium. Tucson, USA. 2007: 149-160

[10] Baruah S, Fisher N. Global deadline-monotonic scheduling of arbitrary-deadline sporadic task systems//Proceedings of the 11th International Conference on Principles of Distributed Systems. Berlin, Germany, 2007: 204-216

[11] Sun Y, Lipari G. Response time analysis with limited carry-in for global earliest deadline first scheduling//Proceedings of the 36th Real-Time Systems Symposium. San Antonio, USA, 2015: 130-140

[12] Chattopadhyay B, Baruah S. Limited-preemption scheduling on multiprocessors//Proceedings of the 22nd International Conference on Real-Time Networks and Systems. Versaille, France, 2014: 225-235

[13] Thekkilakattil A, Baruah S, Dobrin R, et al. The global limited preemptive earliest deadline first feasibility of sporadic real-time tasks//Proceedings of the 26th Euromicro Conference on Real-Time Systems. Madrid, Spain, 2014: 301-310

[14] Guan N, Stigge M, Yi W, et al. New response time bounds for fixed priority multiprocessor scheduling//Proceedings of the 30th IEEE Real-Time Systems Symposium. Washington, USA, 2009: 387-397

[15] Davis R I, Burns A. Priority assignment for global fixed priority pre-emptive scheduling in multiprocessor real-time systems//Proceedings of the 30th IEEE Real-Time Systems Symposium. Washington, USA, 2009: 398-409

[16] Elliott G A, Anderson J H. An optimal *k*-exclusion real-time locking protocol motivated by multi-GPU systems. Real-Time Systems, 2013, 49(2): 140-170

[17] Elliott G A, Ward B C, Anderson J H. GPUSync: A framework for real-time GPU management//Proceedings of the IEEE 34th Real-Time Systems Symposium. Vancouver, Canada, 2013: 33-44

[18] Marinho J, Nélis V, Petters S M, et al. Limited pre-emptive global fixed task priority//Proceedings of the IEEE 34th Real-Time Systems Symposium. Vancouver, Canada, 2013: 182-191

[19] Yao G, Buttazzo G, Bertogna M. Comparative evaluation of limited preemptive methods//Proceedings of the 15th Conference on Emerging Technologies and Factory Automation. Bilbao, Spain, 2010: 1-8

[20] Davis R I, Burns A. Improved priority assignment for global fixed priority pre-emptive scheduling in multiprocessor real-time systems. Real-Time Systems, 2011, 47(1): 1-40

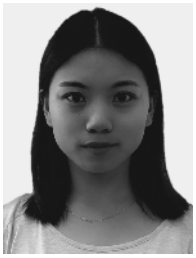
[21] Davis R I, Burns A, Marinho J, et al. Global fixed priority scheduling with deferred pre-emption//Proceedings of the 19th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications. Taipei, China, 2013: 1-11

[22] Thekkilakattil A, Davis R I, Dobrin R, et al. Multiprocessor fixed priority scheduling with limited preemptions//Proceedings of the 23rd International Conference on Real Time and Networks Systems. Lille, France, 2015: 13-22

[23] Bertogna M, Baruah S. Limited preemption EDF scheduling of sporadic task systems. IEEE Transactions on Industrial Informatics, 2010, 6(4): 579-591

[24] Short M. Improved schedulability analysis of implicit L tasks under limited preemption EDF scheduling//Proceedings of the 16th Conference on Emerging Technologies & Factory Automation. Toulouse, France, 2011: 1-8

[25] Lee J, Shin K G. Controlling preemption for better schedulability in multi-core systems//Proceedings of the 33rd IEEE Real-Time Systems Symposium. San Juan, USA, 2012: 29-38



HAN Mei-Ling, Ph. D. candidate. Her main research interests focus on the embedded real-time systems, especially the real-time scheduling on multiprocessor systems.

DENG Qing-Xu, Ph. D., professor. His main research interests include cyber-physical systems, embedded, and real-time systems.

ZHANG Tian-Yu, Ph. D. candidate. His main research interests focus on the embedded real-time systems.

FENG Zhi-Wei, Ph. D. candidate. His main research interests focus on the embedded real-time systems.

LIN Yu-Han, Ph. D. candidate. His main research interests focus on the embedded real-time systems.

Background

Most studies of real-time systems concentrated on the time constraint and omitted the overheads caused by preemption and migration. However, the overheads caused by preemption cannot be omitted. The run-time overheads analysis of embedded real-time multiprocessor systems is an important problem of the real-time scheduling field. Using limited preemption to reduce the runtime overheads of mature scheduling algorithm is popular. There are already many technologies combining the global fixed priority scheduling with limited preemption. However, there is less work in combining the global earliest deadline first (G-EDF) scheduling method with limited preemption.

This work proposed a method combining the G-EDF scheduling with limited preemption to reduce the number of preemptions of fully preempted G-EDF. The final purpose of this work is reducing the overheads caused by preemption and tasks migration. The schedulability of this work is not worse than G-EDF.

This work is founded by "Design and Analysis Research on Virtualization-Based Multi-Core Real-Time Systems" project. The purpose of this project is designing and analyzing the techniques about the multiprocessor interrupt management and hierarchical scheduling on multiprocessor systems and so on. Our research results in this direction are: the paper titled "Transforming Real-Time Task Graphs to Improve Schedulability" published in *RTCSA*, the paper titled "Start Time Configuration for Strictly Periodic Real-Time Task Systems" published in *JSA*.

This work also is founded by "Real-Time System Modeling and Energy Management for Safety-Critical Technology Research on CPS". This project intends to investigate flexible and expressive task execution models for cyber-physical systems. The targeting systems are safety-critical systems with hard real-time requirements. We will firstly identify and characterize different models and explore their efficiency, effectiveness, and use cases while performing the design and analysis for the work packages. Our research results in this direction are: the paper titled "Modular Performance of Energy Harvesting Real-Time Networked Systems" published in RTSS, the paper titled "Bounding Carry-in Interference to Improve Fixed-Priority Global Multiprocessor Scheduling Analysis" published in *RTCSA* and so on.