软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2020,31(11):3657−3670 [doi: 10.13328/j.cnki.jos.005839]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

一种基于动态需求边界的混合关键级作业调度算法*

曾理宁[1], 徐 成[1], 李仁发[1], 杨 帆[2], 徐洪智[1,3]

[1](嵌入式与网络计算湖南省重点实验室(湖南大学),湖南 长沙 410082)

[2](中南林业科技大学 计算机与信息工程学院,湖南 长沙 410004)

[3](吉首大学 软件学院,湖南 张家界 427000)

通讯作者: 徐成, E-mail:cheng_xu@yeah.net

摘 要: 把具有不同重要性的功能集成到一个共享平台上的混合关键级系统,是当前嵌入式系统发展的主要趋势之一.已有的混合关键级调度理论为了保证高关键级作业的完成,大多不支持关键级向下切换,在系统进入高关键级后直接放弃低关键级作业的执行,这对系统中作业集的整体完成率有负面影响.为了应对这一问题,把需求边界分析理论扩展到混合关键级作业系统中,提出了作业的动态需求边界函数,以矢量的形式记录系统在运行时需求边界函数的动态变化,并相应地提出了作业的混合关键级松弛时间与系统关键级松弛时间的概念.在此基础上,提出了一种基于动态需求边界的混合关键级作业调度算法 CSDDB(criticality switch based on dynamical demand boundary).该算法选择具有最小松弛时间的关键级作为执行关键级,在保证高关键级作业可调度的情况下,充分利用系统资源,尽可能地满足低关键级作业的执行.应用随机生成的任务集进行仿真实验,结果表明,与已有算法相比,CSDDB 在系统关键级的保证与作业集整体完成率方面比现有算法有 10%以上的提升.

关键词: 混合关键级;关键级切换;实时调度;需求边界函数;松弛时间

中图法分类号: TP316

中文引用格式: 曾理宁,徐成,李仁发,杨帆,徐洪智.一种基于动态需求边界的混合关键级作业调度算法.软件学报,2020,31(11):3657−3670. http://www.jos.org.cn/1000-9825/5839.htm

英文引用格式: Zeng LN, Xu C, Li RF, Yang F Xu HZ. Scheduling algorithm for mixed-criticality jobs based on dynamical demand boundary. Ruan Jian Xue Bao/Journal of Software, 2020,31(11):3657−3670 (in Chinese). http://www.jos.org.cn/1000-9825/5839.htm

Scheduling Algorithm for Mixed-criticality Jobs Based on Dynamical Demand Boundary

ZENG Li-Ning[1], XU Cheng[1], LI Ren-Fa[1], YANG Fan[2], XU Hong-Zhi[1,3]

[1](Key Laboratory for Embedded and Network Computing of Hu'nan Province (Hu'nan University), Changsha 410082, China)

[2](College of Computer and Information Engineering, Central South University of Forestry and Technology, Changsha 410004, China)

[3](College of Software, Jishou University, Zhangjiajie 427000, China)

Abstract: An important trend in embedded system is integrating functions with different level of importance into a sharing hardware platform, which is called mixed-criticality system. Most of the existing mixed-criticality theory did not support switching the system criticality from high to low in order to guarantee the jobs with higher criticality, which is not good for the overall performance of the

* 基金项目: 国家自然科学基金(61772185, 61672217, 61173036); 国家重点研发计划(2016YFB0200405); 国家高技术研究发展计划(863)(2012AA01A301-01); 湖南省自然科学基金(2019JJ50996)

Foundation item: National Natural Science Foundation of China (61772185, 61672217, 61173036); National Key Research and Development Program of China (2016YFB0200405); National High Technology Research and Development Program of China (863) (2012AA01A301-01); Natural Science Foundation of Hu'nan Province (2019JJ50996)

收稿时间: 2018-04-18; 修改时间: 2018-09-20; 采用时间: 2019-03-25; jos 在线出版时间: 2019-11-06

CNKI 网络优先出版: 2019-11-06 11:49:10, http://kns.cnki.net/kcms/detail/11.2560.TP.20191106.1148.004.html

system. To deal with this problem, this paper expands the traditional demand boundary analysis theory to the mixed-criticality systems, presenting the concept of dynamical demand boundary for mixed-criticality jobs, which represents the dynamical demand of jobs in run-time as a vector. And then, based on the concept of slack time for mixed-criticality jobs and the criticality of system, the paper presents an algorithm CSDDB (criticality switch based on dynamical demand boundary). The algorithm chooses the criticality with the minimum slack time as the execution criticality of the system to take full advantage of system resources and to guarantee the execution of jobs with lower criticality without affecting the schedulability of high criticality jobs. Experiments with randomly generated workload show that CSDDB makes more than 10% of progress in guaranteeing the system criticality and the completion of jobs set compared with the existing research.

Key words: mixed-criticality; criticality switch; real-time scheduling; demand boundary function; slack time

实时嵌入式系统的一个发展趋势,是把具有不同关键等级的功能集成到一个共享的硬件平台上.这种具有多种关键等级的系统被称为混合关键级系统.其中,关键等级是系统为了防止功能运行失败而提供的安全保障机制,其高低取决于功能失效时系统可能受到的后果的严重性.在混合关键级系统中,更高的关键级通常对应着优先的系统资源,如更多的预留执行时间、更优先的计算资源分配等.目前,包括汽车电子系统、飞行器等领域在内的多数嵌入式系统都在朝着混合关键级系统的方向发展,以满足系统在成本、空间、重量、功耗、热量等方面的要求.

混合关键级系统中的一个基本问题是:如何在计算资源有限的前提下,保证系统中具有更高关键等级的功能完成.显然,传统的实时调度理论由于没有考虑系统作业的关键级属性,无法解决这一问题.因此,混合关键级系统调度开始成为一个重要的研究方向.文献[1]最先提出了一种混合关键级模型,随后被不断深入研究[2−4].该模型改变了传统实时调度理论中作业的最坏执行时间(worst case execution time,简称 WCET)的描述,认为一个作业的 WCET 并非一个固定的值,而是一个与关键等级相关的矢量,不同的关键级对应不同保守程度的 WCET 估计.其中,更高的关键等级意味着更保守的估计及更大的 WCET 值.在运行时,如果某作业的实际运行时间超过了相应较低关键级下的 WCET,则称该作业在该关键级过载.当过载发生时,系统关键级将提升,以应对更高关键级下的更大 WCET.在这种模型下,很多算法被提出[5−14].其中,典型的如 OCBP 算法[8]、EDF-VD 算法[9]以及 zero-slack 算法[10]等.这些算法都运用了以下基本策略.

- 系统初始化运行在最低的关键级下,若所有作业的执行都没有过载,系统始终保持在当前关键级内;
- 当且仅当系统检测到在当前关键级下有作业过载,系统切换至更高的关键级;
- 当系统运行于某关键级时,所有自身关键级低于系统关键级的作业都被丢弃;
- 系统不会主动切换到更低的关键级,直到系统空闲,系统关键级才被重新初始化为最低等级.

然而,上述的基本策略在实际应用中可能造成以下两个问题.

(1) 当系统处于高关键级时,低关键级作业总是被简单放弃.但低关键级不等于不重要,丢弃低关键级作业会不可避免地造成系统功能的缺失和用户体验的下降.

(2) 系统关键级的向下切换仅发生在系统空闲时,这使得在系统空闲之前,所有自身关键级低于当前系统关键级的作业都无法正常执行.

针对上述两个问题,研究者开始关注混合关键级调度中低关键级作业的执行,对混合关键级系统的研究重点逐步转变为如何在保证高关键级作业完成的前提下,尽可能好地执行低关键级作业.文献[15,16]提出了一种补偿协议 Bailout Protocol(BP),该协议改变了双关键级系统中系统的关键级行为模式,定义了多种系统状态及其转换条件,通过在一定条件下主动调整关键级,在不损害高关键级作业执行的前提下,尽可能为低关键级作业提供运行时间.文献[17]提出了一种运行时机制,可以发现并使低关键级任务在弹性活动模式中安全的切换.文献[18]通过动态地执行预算管理,回收作业在高关键级下的冗余执行时间,并分配给低关键级作业.文献[19]从概率角度分析了作业的实际执行时间,并通过尽可能延后低关键级向高关键级的切换时间,来为低关键级作业提供更高的服务保证.文献[20]提出了一种以截止时间为关键参数的混合关键级调度算法,通过对任务的截止时间进行动态调整,优化低关键级任务的调度.然而,这些方法在系统关键级高时,通常充分考虑最坏情况,使关键级向下迁移的条件极为严格,仍然影响了低关键级作业的执行.

另外,混合关键级作业调度已经被扩展到了多处理器平台.但多处理器上的混合关键级作业调度通常包括作业的划分以及单处理器上的调度等两个部分.其中,单处理器上的调度算法是调度的基本策略,直接影响作业在多处理器上的划分.因此,研究单处理器上的调度策略,对于主流的多处理器平台上混合关键级调度的研究,依然具有重要的意义.

本文以混合关键级作业系统为研究对象,提出了面向混合关键级作业的动态需求边界函数,以矢量形式表述作业在各关键级下的需求边界,充分考虑作业在运行时的实际执行对需求边界的动态影响.在动态需求边界的基础上,提出以关键级松弛时间为主要参数的关键级切换算法.通过对需求边界的动态更新,提前对系统关键级进行切换,从而在不影响高关键级作业运行的情况下,尽可能地为低关键级作业提供更多的执行机会,以提升任务集的整体执行率.

本文第 1 节介绍混合关键级作业系统模型和相关的定义.第 2 节引入传统实时调度理论中的需求边界函数,并针对混合关键级作业系统的特征,重新定义混合关键级作业的动态需求边界函数,以及基于动态需求边界的关键级松弛时间.第 3 节提出基于动态需求边界的关键级切换算法(criticality switch based on dynamic demand boundary,简称 CSDDB).第 4 节对 CSDDB 算法进行仿真和分析.最后,第 5 节对全文进行总结.

1 系统模型与相关定义

在混合关键级作业系统中,一个作业是一段有限的可执行的代码.在一个由 n 个互不关联的作业构成的具有 L 个关键等级的混合关键级作业集 $\tau=\{J_1,J_2,\ldots,J_n\}$ 中,每一个作业 J_i 都可以表示为四元组 $J_i=(a_i,d_i,\chi_i,C_i)$.其中,

- a_i 是作业的到达时间,即作业从时刻 a_i 开始就绪可执行.
- d_i 是作业的截止期限.
- $\chi_i(1\leqslant\chi_i\leqslant L)$ 是作业的自身关键等级,χ_i 越大,关键等级越高.当系统关键级 $\chi>\chi_i$,系统不要求作业 J_i 完成.相应地,当 $\chi\leqslant\chi_i$,J_i 必须完成.
- C_i 是一个矢量,表示为:$C_i=(C_i(1),C_i(2),\ldots,C_i(L))$,其中,$C_i(k)$ 表示作业 J_i 在关键级 k 上的 WCET.$C_i(k)$ 满足以下两条性质:
  (1) $C_i(k)$ 随 k 单调非减,如果 $k_1\leqslant k_2$,则 $C_i(k_1)\leqslant C_i(k_2)$.这是因为作业的关键级代表对作业 WCET 估计的保守程度,作业的关键级越高,WCET 的估计就越保守,对应的值越大.
  (2) 当 $k\geqslant\chi_i$ 时,$C_i(k)=C_i(\chi_i)$.即当系统的关键级超过了作业自身的关键等级,作业的 WCET 不再随系统关键级的增加而增加.

在一个混合关键级作业系统中,定义 c_i 为作业 J_i 在时间片 $[a_i,d_i]$ 之内完成所使用的实际执行时间.需要说明的是,c_i 是一个与作业在运行时的实际情况相关的值,无法被提前预知,当且仅当该作业的代码已经执行完毕,作业向系统发出完成信号时,c_i 的值才能够确定.在本文的模型中,作业在自身关键级下的执行时间总是足够保守,因此 $c_i\leqslant C_i(\chi_i)$ 总是成立的.

定义 1. 混合关键级作业集 τ 的一个实例 I 是当该作业集被调度时,每一个作业获得的实际执行时间 $(c_1,c_2,\ldots,c_n)$ 的集合.

定义 2. 在混合关键级作业系统的一个实例 I 中,作业 J_i 的执行关键级是满足 $c_i\leqslant C_i(k)$ 的关键级 k 的最小值.

定义 3. 在混合关键级作业系统的实例 I 中,系统关键级 χ_s 是指对于所有满足 $\chi_i\geqslant\chi_s$ 的作业 J_i,都能在时间片 $[a_i,d_i]$ 内完成,且满足执行时间 $c_i\leqslant C_i(\chi_s)$ 的关键级 χ_s 的最小值.如果不存在这样的关键级 χ_s,那么该实例是不可调度的.

通过以上定义可知,作业的实际执行时间是与实际运行情况相关的.因此,随着实际执行情况的变化,作业集的实例也会随之变化.因此,实例是无穷的.混合关键级作业调度的目标就是:寻找某种策略,使得作业集产生的任意实例,都能够获得一个具有系统关键级 χ_s 的正确调度,且 χ_s 的值尽可能地小,使系统能保证更多关键级下的作业完成.

本文中的作业都运行在可抢占单处理器上,采用经典的 EDF(earlist deadline first)算法作为基本调度策略.

EDF 总是调度作业集中具有最早截止期限的作业,已经被证明为单处理器上最优的作业调度方法,其可调度的条件是系统利用率 $U\leqslant 1$.

2 混合关键级作业的动态需求边界

2.1 动态需求边界的定义

在传统实时调度理论中,需求边界函数(demand boundary function,简称 DBF)是测试作业集可调度性的重要方法,主要思想是:通过计算作业在一段时间内的最大需求(需求边界),来判断该作业在其时限前是否可以获得足够多的执行时间.在传统实时调度理论中,作业的需求边界由两个部分组成:(1) 作业在时间片内可能需要的最长执行时间,即作业的 WCET;(2) 其他所有可能阻塞该作业执行的作业的执行时间.显然,如果在一段时间片内,作业的需求边界小于该时间片的长度,那么该作业是可调度的.

但在混合关键级中,由于以下原因,需求边界函数无法直接应用.

(1) 模型的改变.传统实时作业模型中,作业的 WCET 是一个数值,表示作业的执行时间的最大可能值.但混合关键级作业的WCET是与关键等级相关的矢量,在不同关键级下有不同的WCET.因此,在考虑混合关键级作业的需求边界时,必须考虑到关键等级的影响.

(2) 关键级阻塞.在混合关键级作业系统中,作业除了被具有更高优先级的作业阻塞之外,还可能被具有更高关键等级的作业阻塞,以保证高关键级作业的优先完成.因此,相对于传统实时系统中的需求边界,混合关键级作业的阻塞情况更为复杂.

(3) 关键级切换.在混合关键级系统中,当系统关键级出现切换,调度的目标随之变化.典型的一种情况是:当系统关键级升高,系统可能丢弃所有低关键级作业,使待调度的作业集发生变化.特别地,关键级的切换总是发生在运行时,这使得混合关键级作业的需求边界具有了动态的特性.

因此,本文考虑混合关键级系统的特性,构建了适用于混合关键级作业系统的动态需求边界函数(dynamic demand boundary function,简称 DDBF).DDBF 在传统需求边界的基础上,首先定义了作业在不同关键级上的不同需求边界;然后考虑了需求边界在运行时可能受到的动态影响,把 DDBF 定义为一个与系统执行情况相关的动态值.本文定义 DDBF 如下.

定义 4. 在混合关键级作业系统中,一个作业 J_i 的动态需求边界函数 $ddbf_i^{\chi}(t)$ 被定义为:当系统处于关键级 $\chi(\chi\leqslant\chi_i)$时,$J_i$ 在时刻 t 的时间需求的上界.

2.2 动态需求边界的构成

根据定义 4,混合关键级作业的动态需求边界需考虑以下几方面的内容.

(1) 作业 J_i 在给定关键级 χ 的执行时间 $e_i^{\chi}(t)$

在给定的关键级 χ 上,在时刻 a_i 之前,作业 J_i 未就绪,作业所需的执行时间为 0.当作业 J_i 在时刻 a_i 就绪,系统要求该作业在 d_i 之前完成.根据定义 4,若作业的自身关键级 χ_i 满足 $\chi_i\geqslant\chi$,J_i 将被分配的执行时间是其在关键级 χ 下的 WCET,即 $e_i^{\chi}(t)=C_i(\chi)$;反之,若 $\chi_i<\chi$,则在关键级 χ 下,J_i 的完成并非系统保证的目标,那么 $e_i^{\chi}(t)=0$.因此,

$$e_i^{\chi}(t)=\begin{cases}C_i(\chi), & \chi_i\geqslant\chi\wedge t\geqslant a_i\\ 0, & \text{otherwise}\end{cases} \tag{1}$$

(2) 作业 J_i 在给定关键级 χ 下的优先级阻塞 $b_i^{\chi}(t)$

本文采用 EDF 算法作为基本调度策略,截止时间早的作业具有更高的优先级.在给定关键级 χ 下,作业 J_i 被另一个作业 J_j 阻塞的条件是:a) 作业 J_j 的关键级 χ_j 不低于给定关键级 χ,因为任何自身关键级低于 χ 的作业都不会被考虑;b) 作业 J_j 具有高于作业 J_i 的优先级,即 $d_j\leqslant d_i$.满足该条件的作业 J_j 最多可能阻塞作业 J_i 的时长为其在系统关键级 χ 下的 WCET,即 $C_j(\chi)$.因此,

$$b_i^{\chi}(t)=\sum_{J_j:d_j\leqslant d_i\wedge\chi_j\geqslant\chi}C_j(\chi) \tag{2}$$

(3) 运行时的动态影响 $d_i^{\chi}(t)$

在运行时,若某作业 J_j 在时刻 t 之前执行,那么其执行关键级χ^*会给作业 J_i 的需求边界带来动态的影响.在给定关键级χ上,需要考虑两种情况.

1) $\chi^*>\chi$.对于作业 J_j,当$\chi^*>\chi$时,$C_j(\chi^*)>C_j(\chi)$.因此,对于给定关键级χ,J_j 的执行消耗了更多的时间,增加值为 $C_j(\chi^*)-C_j(\chi)$.该部分增加的时间会使作业 J_i 的需求增加,因此也应该计算到其需求边界中.

2) $\chi^*<\chi$.对于作业 J_j,当$\chi^*<\chi$时,$C_j(\chi^*)<C_j(\chi)$.因此,对于给定关键级χ,J_j的执行事实上并未消耗在给定关键级χ下的预期执行时间 $C_j(\chi)$.因此,其"提前"完成降低了作业J_i的需求,降低的时长为 $C_j(\chi)-C_j(\chi^*)$,即需求上界的边界增量为 $C_j(\chi^*)-C_j(\chi)$.

事实上,任何在时刻 t 之前执行的作业都可能因为关键级的变化给作业 J_i 在时刻 t 的需求边界带来影响.定义 τ_t^* 为在时刻 t 之前执行的作业的集合,那么系统在其他关键级上的影响因子可以表述为

$$d_i^{\chi}(t)=\sum_{J_j\in\tau_t^*\wedge\chi_j\geqslant\chi}(C_j(\chi^*)-C_j(\chi)) \tag{3}$$

这里,$d_i^{\chi}(t)$ 的值总是与在时刻 t 之前执行的其他作业的执行关键级相关,只能在运行时更新.

(4) 到达时间的影响

最后,还需要考虑作业 J_i 的到达时间 a_i.当作业 J_i 在时刻 a_i 就绪,如果系统处于空闲状态,说明其他可能阻塞 J_i的作业都已经执行完成,J_i可以直接执行,因此,J_i在时刻 a_i的需求上界为 $t+e_i^x(t)$;否则,系统中还有其他作业执行,此时,J_i 的需求上界为 $e_i^x(t)+b_i^x(t)+d_i^x(t)$.

综合考虑上述情况,当系统关键级为χ时,作业 J_i 在时刻 t 的需求边界可以表述为

$$ddbf_i^{\chi}(t)=e_i^x(t)+\max(b_i^x(t)+d_i^x(t),a_i) \tag{4}$$

2.3 动态需求边界示例

为了说明混合关键级作业的需求边界函数,考虑以下例子.

例 1:给定一个具有 2 个关键级的混合关键级作业系统,见表 1.

Table 1 An example of mixed-criticality jobs system

表 1 混合关键级作业系统实例

J_i	a_i	d_i	χ_i	$C_i(1)$	$C_i(2)$
J_1	1	3	1	1	1
J_2	0	5	2	2	4

根据公式(4),可以计算出两个作业在不同关键级下的需求边界函数,如图 1 所示.

从图 1 可以看出,一个作业的需求边界是关于时间的阶梯函数.以 $ddbf_2^1(t)$ 为例,作业 J_2 在时刻 0 就绪,需要在时间片[0,4]内被分配 1 个时间单位的执行时间,因此,$ddbf_2^1(0)=e_2^1(0)=2$;在时刻 1,作业 J_1 就绪,由于 J_1 的时限 3 早于 J_2 的时限 5,根据 EDF 算法思想,J_1 的优先级高于 J_2,因此 J_1 可以阻塞 J_2,$ddbf_2^1(1)=e_2^1(1)+b_2^1(1)=2+C_1(1)=3$.相应地,在时刻 1,$J_1$ 就绪,没有其他作可以阻塞其执行,但需要考虑其到达时间的影响,其需求边界为

$$ddbf_1^1(1)=e_1^1(1)+\max(b_1^1(1)+d_1^1(1),1)=2.$$

然后,如图 2 所示,假设在时刻 3,作业 J_2 已经执行了 2 个时间单位,但未声明完成,即 J_2 发生了过载,需要系统为其分配比 $C_2(1)$更多的执行时间.此时,$d_2^1(3)=C_2(2)-C_2(1)=4-2=2$.因此,$ddbf_2^1(t)$ 在时刻 3 发生了动态变化,增加了 2 个单位.对于作业 J_1,由于 $d_2>d_1$,J_2 的优先级低于 J_1,$ddbf_1^1(t)$ 的值不会因为 J_2 的过载发生变化.

动态需求边界函数对于判断某个作业在给定关键级上的可调度性是有效的.当 $ddbf_i^{\chi}(t)=t$,作业在时刻 t 可以获得足够的执行时间以完成,如果时刻 $t\in[a_i,d_i]$,意味着该作业可以在时限之前获得足够的执行时间,该作业是可调度的.在例 1 中,对于作业 J_1,$ddbf_1^1(2)=2$,这意味着如果系统关键级为 1,J_1 将在时刻 2 之前获得足够多的执行时间.显然,当系统关键级为 1 时,J_1 是可调度的.对于作业 J_2,从图 1 可知,由于 $ddbf_2^1(3)=3$,如果系统始终

保持在关键级 1 下执行,那么 J_2 最晚将在时刻 3 之前获得 2 的执行时间并完成.当 J_2 发生过载,其执行时间增加到 4, $ddbf_2^1(t)$ 在时刻 3 之后增加到 5.但依然能在它的时限时刻 5 满足: $ddbf_2^1(5)=5$,也就是说,即使 J_2 发生了过载,它也能在其时限 5 之前获得足够的执行时间,依然是可调度的.

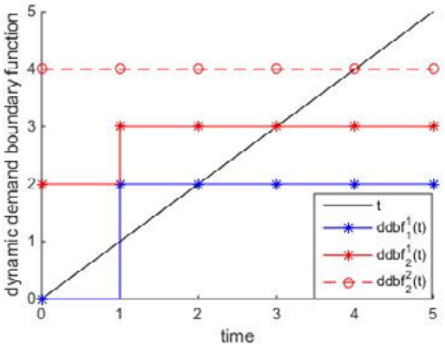

Fig.1 Initial value of dynamic demand boundary function in Example 1

图 1 例 1 中动态时间需求函数的初始值

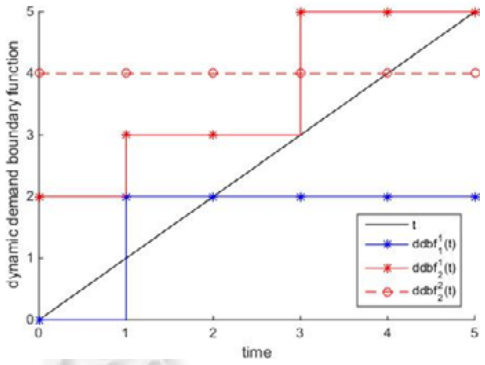

Fig.2 Dynamic demand boundary function when overload happens in Instant 3

图 2 在时刻 3 发生过载时的动态时间需求函数

2.4 关键级松弛时间与可调度性

定理 1. 在系统关键级 χ 下,混合关键级作业系统中的某个作业 J_i 能够被调度,当且仅当:

$$ddbf_i^{\chi}(t)\leqslant t \tag{5}$$

在时间片 $[a_i,d_i]$ 中有解,且满足 $dbf_i^{\chi}(t)=t$ 的时刻是作业 J_i 完成的最晚时刻.

证明:根据混合关键级作业的动态需求边界函数的定义,作业 J_i 在给定的关键级 χ 下的需求边界是当系统的执行关键级不超过 χ 时,J_i 需要完成执行所需要的时间的上界.当 $ddbf_i^{\chi}(t)\leqslant t$,系统在时刻 t 提供的时长不小于 J_i 所需的时长.由于本文采用的基本调度算法为 EDF,当还有作业未完成时,处理器永不空闲,因此在时刻 t,作业 J_i 一定能够得到足够的执行时间,如果 $t\in[a_i,d_i]$,那么 J_i 是可调度的.

定义 t_i^{χ} 为当 $ddbf_i^{\chi}(t)=t$ 的时刻,在 EDF 调度策略中,如果所有作业的实际执行时间都是当前关键级下的 WCET,那么作业 J_i 将在时刻 t_i^{χ} 完成执行.也就是说,如果系统关键级保持在 χ, t_i^{χ} 是 J_i 完成的最晚时刻.

综上所述,定理 1 正确. □

考虑 t_i 与时限 d_i 的关系,如果 $t_i^{\chi}>d_i$,那么 J_i 无法在 d_i 之前得到足够的执行时间,是不可调度的;反之,如果 $t_i^{\chi}\leqslant d_i$, J_i 是可调度的.

定义 5. 在系统关键级 χ 下,作业 J_i 的松弛时间 $s_i^{\chi}=d_i-t_i^{\chi}$,其中, t_i^{χ} 是 $ddbf_i^{\chi}(t)=t$ 的最小解.

如果作业 J_i 的松弛时间 $s_i^{\chi}\geqslant 0$,那么在系统关键级始终不超过 χ 的情况下,作业 J_i 是可调度的,并且在其时限 d_i 之前,还有至少 s_i^{χ} 个时间单位冗余.即系统在运行过程中,如果 J_i 的需求边界的动态部分不超过 s_i^{χ} 个时间单位,那么 J_i 总是可调度的.

考虑整个作业集,对于所有自身关键级小于系统关键级 χ 的作业,在 χ 下的松弛时间是没有意义的.对于自身关键级不小于 χ 的作业,如果所有作业的松弛时间都不小于 0,那么如果系统运行在关键级 χ,且没有出现过载,作业集是可调度的.因此,可以定义系统在给定关键级下的松弛时间.

定义 6. 混合关键级作业系统 τ 在关键级 χ 下的松弛时间为

$$S^{\chi}=\min_{J_i\in\tau:\chi_i\geqslant\chi}\{s_i^{\chi}\} \tag{6}$$

根据定义 3 与定义 6,在关键级 χ 下,如果 $S^{\chi}\geqslant 0$,那么如果系统关键级为 χ,作业集是可调度的;反之,当 $S^{\chi}<0$,那么作业集在 χ 下不可调度,系统必须提升关键级以才能保证系统的正确性.

需要说明的是,作业的松弛时间 s_i^χ 和系统的松弛时间 S^χ都是动态的,都会根据系统关键级的变化而更新.以例 1 中的作业集为例,在时刻 3 之前, $ddbf_1^1(2)=2$,因此 $t_1^1=2, s_1^1=d_1-t_1^1=1, s_2^1=d_2-t_2^1=2$,故 $S^1=\min(s_1^1,s_2^1)=1$.相应地, $S^2=s_2^2=1$.此时,无论系统关键级是 1 或 2,在保证系统正确的前提下,都还有 1 个时间单位的冗余时间.在时刻 3,作业 J_2 过载,使作业的边界和松弛时间都发生动态了变化.作业 J_1 由于具有更高的优先级,不会因 J_2 的过载而改变其需求边界与松弛时间.但对于作业 J_2,在关键级 1 下, $ddbf_2^2(5)=5, t_2^1=5, s_2^1=d_2-t_2^1=0$,因此 $S^1=\min(s_1^1,s_2^1)=0$,这意味着在时刻 3,如果系统按照关键级 1 执行,系统可以保持正确,但没有冗余时间;在关键级 2,由于作业 J_2 的执行并未超过其在关键级 2 下的 WCET,因此 S^2=1 保持不变,如果系统按照关键级 2 执行,系统保持正确的前提下还有 1 个时间单位的冗余.

系统在各关键级下的松弛时间,表示系统能否保持正确性以及在当前时刻保持该关键级下的正确性之外的冗余时间.在混合关键级作业系统的调度中,系统在各关键级下的松弛时间可以作为调度的重要参数.

3 基于动态需求的关键级切换算法

本节根据混合关键级作业系统中作业在给定关键级下的动态需求边界以及松弛时间的定义,提出了一种新的混合关键级作业调度算法:基于动态需求边界的关键级切换算法(criticality switch based on dynamic demand boundary,简称 CSDDB).对于一个给定的混合关键级作业集τ,CSDDB 的处理方式是:在实时更新的作业动态需求边界及其对应的关键级松弛时间的基础上,总是在所有不小于 0 的关键级松弛时间中,选择最小值所对应的关键级作为系统的执行关键级;如果出现多个关键级松弛时间同时最小的情况,选择其中最高的关键级作为系统的执行关键级.在运行的过程中,按照 EDF 的策略调度.

3.1 算法描述

首先定义系统的执行关键级χ_s.在 CSDDB 中,选择χ_s作为系统的执行关键级,意味着在接下来的时间片中,系统仅仅调度作业集中满足$\chi_i \geqslant \chi_s$ 的作业.χ_s 的选择基于各关键级的松弛时间.显然,当某个作业集上的松弛时间 S^χ<0,那么系统运行于该关键级上是不可调度的;反之,当 $S^\chi \geqslant 0$,松弛时间越短就意味着该关键级上的容错能力越低,因此应该提前执行.值得注意的是,系统的事件如作业到达、作业完成、作业超载等,都可能引起各作业的动态需求边界的变化,进而造成各关键级的松弛时间变化,影响到系统执行关键级的选择.因此,设计 CSDDB 算法如算法 1 所示.

算法 1. 基于动态需求边界的关键级切换算法(CSDDB).

输入:作业集τ.

输出:τ的一个调度.

1. 对所有的作业 J 以及所有的关键级χ,计算 $ddbf_i^\chi(t)$ 以及松弛时间 s_i^χ;
2. 对所有的关键级χ,计算关键级松弛时间 S^χ;
3. 系统关键级初始化:χ_s=0;
4. **FOR** t=0 to t=max(d_i) **do**
5. $\chi_s = \chi : S^{\chi_i} \geqslant 0 \wedge \min(S^{\chi_i})$;
6. **IF** (χ_s==0) **return** False;
7. J_e=J_i:$t \in [a_i,d_i] \wedge \min(d_i)$;
8. *execute*(J_e);
9. *EventHandle*(·);
10. t++;
11. **END FOR**;

如算法 1 所示,CSDDB 的算法分为静态部分与运行时部分.其中,静态部分为伪代码的第 1 行~第 3 行.对于

一个已知的作业集,可以根据公式(1)~公式(4)计算各作业的动态需求边界函数,其中,$d_i^\chi(t)$的值取 0.相应地,可以根据公式(5)计算作业的松弛时间,根据公式(6)计算关键级的松弛时间.从伪代码第 4 行开始进入运行时部分.在每一个时间点,系统总是选取所有不小于 0 的关键级松弛时间中的最小值所对应的关键级作为系统关键级(第 5 行).选择所有自身关键级不小于系统关键级的作业中具有最早时限的作业执行(第 7 行),如果这样的关键级不存在,即所有的关键级松弛时间都小于 0,那么系统是不可调度的(第 6 行).在第 9 行,函数 *EventHandle*(·)是算法中的核心函数,主要用于处理系统中的事件,如作业执行、作业完成、作业超时等,并根据系统的事件,更新作业的需求边界与松弛时间.*EventHandle*(·)如算法 2 所示.

算法 2. 事件处理函数 *EventHandle*.

输入:作业集τ,当前时刻 t.

输出:更新作业的动态需求边界和关键级松弛时间.

1. *getEventQueue*(·); //获取事件队列
2. **WHILE** (*EventQueue*≠∅)
3. **IF** (event: J_i在关键级χ_e下执行)
4. **FOR** all $\chi<\chi_e \vee d_k<d_i : d_k^\chi(t)++;$
5. **FOR** all $\chi>\chi_i : d_k^\chi(t)++;$
6. **END IF**;
7. **IF** (event: J_i在关键级χ_e下完成)
8. **FOR** all $\chi \geqslant \chi_e : d_k^\chi(t)+=C_i(\chi_e)-C_i(\chi_i);$
9. **END IF**;
10. **IF** (event: J_i在关键级χ_e下超时)
11. χ_e++;
12. **END IF**;
13. **END WHILE**;
14. **update** all s_i^χ and S^χ;

如算法 2 所示,*EventHandle*(·)函数用于处理系统事件并跟新系统状态.在第 3 行~第 6 行,当作业 J_i在关键级χ_e下执行,所有低关键级、高优先级的需求边界的动态部分会受到影响.同样,由于作业 J_i的执行不会被包括在所有高于其自身关键级χ_i的需求边界中,因此也需要累加.在算法 2 中,χ_e是作业的执行关键级,会根据作业的实际情况不断增加(第 10 行~第 12 行).当作业在关键级χ_e完成,所有高于χ_e的需求边界会降低,增量为$C_i(\chi_e)-C_i(\chi_i)$.当所有事件处理完成,更新所有的松弛时间,用于系统执行关键级的切换.

3.2 算法示例

考虑表 2 中由 3 个作业构成的具有 3 个关键级的作业集的调度.

(1) 在时刻 0,仅作业 J_1就绪.由于 J_1的关键级χ_1=1,因此只考虑 $ddbf_0^1(t)$的取值.由公式(4)可知,$ddbf_1^1(t)=C_1(1)=3$.在当前环境中,当 J_1的时限 5 到达,J_1的松弛时间为 $s_1^1=d_1-t_1^1=3$,显然,此时有 $S^1=s_1^1=3$,系统关键级χ_s=1,选择执行的作业为 J_1.

(2) 在时刻 1,作业 J_2到达.此时,$ddbf_1^1(1)=3+C_2(1)=4, ddbf_2^1(1)=2, ddbf_2^2=3$,因此,$s_1^1=1, s_2^1=2, s_2^2=1$.此时各关键级的松弛时间是:$S^1=\min(s_1^1,s_2^1)=1$,$S^2$=1.系统关键级$\chi_s$=2,选择执行的作业为 J_2.

(3) 时刻 2,作业 J_3到达,若 J_2没有声明完成,J_2的执行关键级增加至 2.此时,

$$ddbf_3^1(2)=\max(C_1(1)+C_2(1),2)+C_3(1)=6, ddbf_3^2(2)=5, ddbf_3^3(2)=a_3+C_3(3)=6.$$

对应地,$S^1=s_3^1=1, S^2=s_3^2=1, S^3=s_3^3=1$,因此,系统关键级$\chi_s$=3,选择执行的作业为 J_3.

(4) 在时刻 3,由于 J_3执行但 $d_3>d_1, d_3>d_2$,因此,J_1和 J_2的需求边界受到影响. $ddbf_1^1(3)=5, ddbf_2^1(3)=3,$

$ddbf_2^2=4$,此时,$S^1=s_1^1=1, S^2=s_2^2=0$,S^3=1,因此,χ_s=2,选择执行的作业为 J_2.

(5) 在时刻 4,作业 J_2 完成,但由于其在执行关键级 2 上执行了一个时间单位,因此,所有在关键级 1 上的边界都必须加上这一个时间单位的动态增量,因此,$ddbf_1^1(4)=6, ddbf_3^1(3)=7, S^1=s_1^1=0$.$S^2$=0 保持不变.但由于 J_2 的自身关键级为 2,因此其执行会影响关键级 3 上的需求边界,$d_3^3(3)=1$,因此,$ddbf_3^3(3)=7$,$S^3=s_3^3=0$.系统关键级χ_s=3,选择执行的作业为 J_3.

(6) 在时刻 5,如果 J_3 完成,那么 J_1 得以在时刻 6 完成;如果 J_3 未能完成,即 c_3>2,那么作业 J_1 将错过时限,但 J_2 与 J_3 都能完成.

Table 2 An example of mixed-criticality jobs set with 3 criticalities

表 2 一个具有 3 个关键级的混合关键级作业集实例

J_i	a_i	d_i	χ_i	$C_i(1)$	$C_i(2)$	$C_i(3)$
J_1	0	6	1	3	3	3
J_2	1	4	2	1	2	2
J_3	2	7	3	2	3	4

3.3 算法复杂度分析

可以从算法的静态部分以及运行时部分来分别说明 CSDDB 算法的复杂度.给定一个具有 n 个作业和 k 个关键级的混合关键级作业集 τ.在静态部分,算法需要计算所有作业在各关键级上的动态需求边界函数.由于 $ddbf_i^x(t)$ 的静态部分只取决于作业本身的执行时间与该作业可能遇到的阻塞,因此,每一个边界的计算都最多需要计算 n 个作业的执行时间,这样的计算需要 $n\times k$ 次,因此,计算需求边界函数的复杂度为 $O(n^2k)$.而无论是作业的松弛时间还是关键级的松弛时间,都只与需求边界函数相关,不会带来更高的复杂度,因此,算法静态部分的复杂度是 $O(n^2k)$.显然,这是多项式级别的.

在运行时,在每一个时刻都需要更新系统的状态.但从算法的描述中可知,系统中的任何一个事件,都最多给每一个需求边界函数带来一次计算.因此,最多需要 $n\times k$ 次;同时,在任何一个时刻可能出现的事件只与作业本身相关,其复杂度为 $O(n)$,因此,运行时部分的复杂度为 $O(n)\times O(n\times k)=O(n^2k)$,显然也是多项式级别的.

因此,CSDDB 的复杂度是多项式级别的.

值得说明的是,在实际运行时,CSDDB 算法的执行代价是可接受的,这是因为:

(1) 在实际应用中,作业的数量 n 和关键级数量 k 是受限的.由于混合关键级作业的运行平台通常以嵌入式平台为主,受限于平台的计算资源以及作业所需计算资源越来越大的实际情况,平台可调度的作业数量 n 是可控的,在绝大多数情况下,$n\leqslant 50$.此外,在现有的混合关键级的应用和标准中,关键级数量通常满足 $k\in[2,7]$,以航空软件认证标准 DO-178 B/C 为例,该标准把航空器中的功能按关键级划分为 A 到 E 的 5 个等级,因此,关键级的数量不会过大的影响算法运行的开销.

(2) 在 CSDDB 中,对动态需求边界函数 $ddbf_i^x(t)$ 在运行前计算,因此无需考虑其对运行时的影响.在运行时,$ddbf_i^x(t)$ 的动态部分需要根据系统运行时发生的事件(如作业到达、作业完成、作业过载等)实时更新,这部分计算是 CSDDB 算法的主要开销.考虑在某时刻 t',算法需要更新 $ddbf_i^x(t')$.需要注意的是,此时需要更新需求边界函数的只有处于未完成状态的作业,而已完成和未到达的作业不受影响.在最坏情况下,即在时刻 t',每一个作业都处于未完成,且每一个作业在 t' 触发事件,算法需要进行 $n^2\times k$ 次动态需求边界函数的更新.但在实际运行中,每个事件发生时需要进行的更新计算次数远小于 $n^2\times k$ 的最坏值.

(3) 由于动态需求函数是一个关于时间点和关键级的离散函数,在运行时,算法为每一个作业 J_i 维护一个二维数组 $db_i[t][\chi]$,表示在时刻 t,作业 J_i 在关键级 χ 下的动态需求边界函数值.当系统中发生事件,引起需求边界函数的动态变化时,需要修改 $db_i[t][\chi]$ 的值.根据公式(4),修改的方式是为当前的值加上由于过载、完成等事件带来的动态变化值,该计算方法并不复杂,不需要占用过多的计算资源.

综上所述,CSDDB 在运行时的开销是可以接受的.

3.4 算法的正确性分析

由于 CSBBD 总是试图在满足高关键级可调度的情况下选择最紧急(松弛时间最少)的关键级作为系统的执行关键级,CSDDB 的运行总是满足以下定理.

定理 2. 如果一个混合关键级作业集τ在其最高关键级下是可调度的,那么该作业集在 CSDDB 下总是能够保证其最高关键级正确.

证明:对于一个已知的混合关键级作业集τ,如果在其最高关键级下可调度,那么根据作业动态需求边界的定义,至少在时刻 0,所有作业的松弛时间都不小于 0,因此$S^{\chi_{\max}}\geqslant 0$.在运行过程中,CSDDB 总是选取关键级松弛时间中不小于 0 的最小值对应的关键级作为系统的执行关键级,且在有多个关键级的松弛时间相同时,取最高关键级作为执行关键级.那么在任意时刻,一旦$S^{\chi_{\max}}=0$,系统关键级$\chi_{\max}$会马上被选为执行关键级,且能够保证即使在最坏情况下,作业集τ在最高关键级$\chi_{\max}$上是正确的. □

引理 1. 对于一个双关键级作业集τ,如果在其高关键级下可调度,那么τ在 CSDDB 上可获得正确的调度.

证明:根据定理 1,双关键级作业集τ在高关键级下是可调度的,也就是说,τ总能够保证高关键级的作业完成.

在保证高关键级作业完成的前提下,CSDDB 通过松弛时间选择系统的执行关键级,把系统切换至更高关键级的时间点尽可能延后,从而为低关键级作业提供更多的执行时间.此外,即使有低关键级作业错过时限,通过关键级松弛时间的更新,系统依然有可能选择低关键级作为系统的执行关键级,从而使低关键级作业的错误率降低. □

我们在下一节对 CSDDB 进行仿真,并与其他已有混合关键级作业调度算法进行比较,以证明其优越性.

4 算法仿真与分析

本文将通过仿真实验来比较 CSDDB 算法与现有的其他混合关键级作业调度算法的性能.实验针对混合关键级作业系统,运行在可抢占单处理器平台上.本文实现了文献[15]中的 Bailout Protocol 算法(BP)、经典的 OCBP 算法[11]以及最基本的关键级即优先级(criticality as priority,简称 CaP)[11]算法,并将其改写成适应于混合关键级作业系统的版本.以这些算法为参照,比较 CSDDB 算法在系统关键级与作业完成率方面的性能,从而评价 CSDDB 的有效性.

4.1 作业集的生成

在仿真中,本文采用文献[11]中生成混合关键级任务集的方法,并将其改变为混合关键级作业集的生成方式.考虑的参数主要包括:

(1) 作业的到达时间 a_i 和时限 d_i.对于一个长度为 T 的时间片,$a_i\in[0,T)$是在整个时间片上的随机值,$d_i\in(a_i,T]$是在 a_i 之后的时间片上的随机值.

(2) 作业的关键等级χ_i.根据文献[10],设关键级$\chi_i\in[1,5]$是一个随机值,且服从概率 $P_\chi=P_{i+1}/P_i$,即选择更高的关键级的概率总是 P_χ.

(3) 作业在自身关键级上的负载最大值 $l_{\max}$.作业 J_i 在自身关键级上的负载 $l_i^{\chi_i}=C_i(\chi_i)/(d_i-a_i)$ 是被设定为在$(0,l_{\max}]$上的随机值,从而作业在自身关键级上的 WCET 也被限定.

(4) 作业在不同关键级的 WCET 之间的比例 P_e.P_e 限定了作业在较低关键级上的 WCET 与较高一个关键级的 WCET 的最大比例,即 $C_i(\chi)/C_i(\chi+1)\leqslant P_e$.在本文的实验中,$P_e$ 取[0.4,0.9]之间的随机数.

(5) 作业超载的概率 P_χ.作业超载至更高一个关键级的概率为 P_χ,即关键级越高,作业在该关键级上运行的概率越低.

(6) 作业集的最大负载 l_s.作业集的最大负载被定义为在时间片[0,T]之内,作业在各关键级上的执行时间的最大值与时长的比例,即 $\max\limits_{\chi\in[1,5]}\left\{\sum\limits_{J_i\in\tau}C_i(\chi)\right\}\Big/T\leqslant l_s$.在本文的实验中,$l_{\max}$ 取值为(0.25,0.85).

作业集生成的方式是:作业集被初始化为空$\tau=\varnothing$,然后依次随机生成混合关键级作业,在满足系统约束的情况下,依次加入作业集中,直到作业集生成完成.其中,添加作业时的约束是:在生成任何一个作业后都计算所有作业的动态需求边界,并依此获得关键级的松弛时间,如果所有的关键级松弛时间都满足 $S^x\geqslant 0$,那么把该作业加入作业集中;否则放弃该作业.作业集生成完成的标志是:连续生成 3 个作业不满足作业添加的约束,或者不满足作业的最大负载.

4.2 实验对比

为了测试CSDDB的性能,本文主要从两个方面进行仿真:(1) 系统关键级;(2) 作业集整体完成率.实验方式如下.

(1) 按照第 4.1 节中描述的方式产生作业集.

(2) 设定超载概率分别为 25%与 50%,在不同的系统总负载条件下分别进行仿真.

(3) 分别使用 CSDDB,BP,OCBP 以及 CaP 算法对同一作业集进行调度,并分别记录结果.

(4) 将上述两个步骤重复执行 20 次,取 20 次结果的平均值作为最终的实验结果.

4.2.1 系统关键级的对比实验

在混合关键级作业集的调度中,系统关键级是调度正确性的重要体现.根据定义 3,系统关键级越低,意味着有更多关键级下的作业能够被正确调度,算法性能越好;反之,系统关键级越高,意味着调度能够保证正确执行的作业越少,算法性能越差.关于系统关键级的对比实验结果如图 3 与图 4 所示.

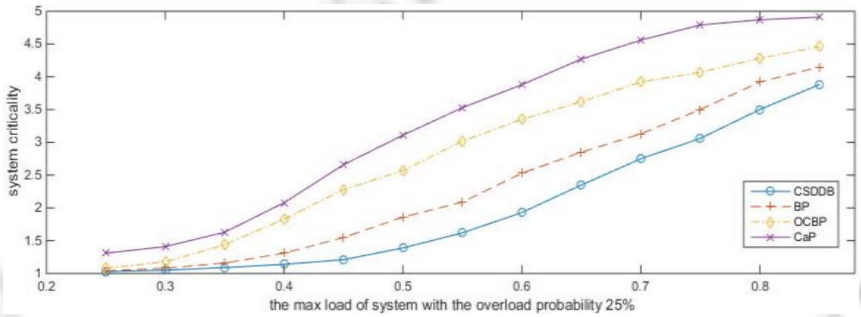

Fig.3 Average system criticality when P_χ=25%

图 3 当 P_χ=25%时的平均系统关键级

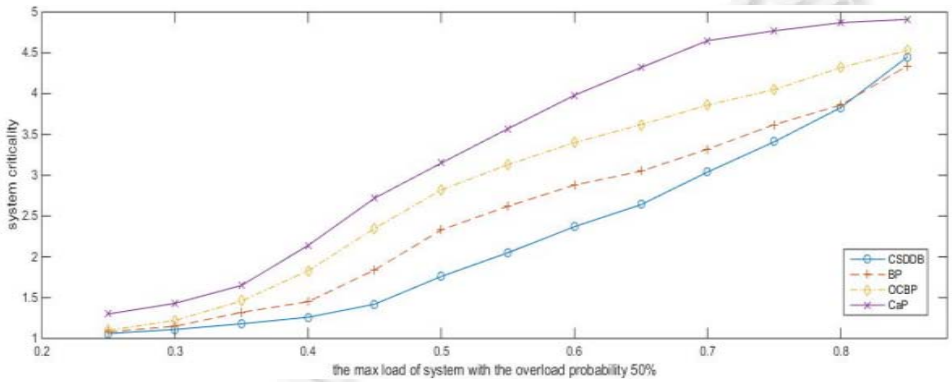

Fig.4 Average system criticality when P_χ=50%

图 4 当 P_χ=50%时的平均系统关键级

在图 3 和图 4 中,x 轴表示系统负载;y 轴表示 20 次运行中,系统关键级的平均值.可以看出,在 4 种算法中,CSDDB 相对于 BP、OCBP 与 CaP,能够使得系统关键级更低.这意味着 CSDDB 在使系统满足更多关键级的正

确性这一目标上具有优势.这是因为在这 4 种算法中,CaP 总是选择具有最高关键级的作业进行调度,一旦作业集负载较高或者超载概率增加,低关键级作业就会被大量放弃,因此,即使在超载概率 25%的情况下,CaP 在负载 85%时的平均系统关键级也达到了接近 5.也就是说,CaP 几乎只能保证具有最高关键级的作业完成.OCBP 提前为作业计算关键级,但当出现超载时,就直接放弃低关键级的作业,因此当超载率较高,作业被大量放弃,系统平均关键级无法降低.BP 提供了关键级回退的机制,这使得系统关键级在很多情况下能够完成从高到低的切换,更多低关键级的作业被完成,从而提升了系统在平均关键级上的性能;但是相对于 CSDDB,BP 关键级切换的过程有一定的滞后,这导致了在系统负载较高时,部分较低关键级作业错过时限.CSDDB 在关键级切换方面更为灵活,系统的状态会直接反映在关键级的松弛时间上,因此整体系能更优.但需要特别注意的是,当 P_χ=50%时,在系统负载超过 0.8 时,CSDDB 的系统平均关键级高于 BP.这是因为当系统负载很高时,BP 不会切换到非常低的关键级,如 1 或 2,但始终能尽量保证关键级 4 或 5 的作业的执行,而 CSDDB 由于总是选择具有最小松弛时间的关键级,导致在某些时刻,系统运行的是更低关键级的作业.而在系统负载很高的前提下,这种情况事实上对系统平均关键级参数是有负面影响的.然而,考虑实际应用中的大多数情况与作业 WCET 的测算方式,P_χ=50%的可能性非常低,因此,CSDDB 在这种情况下的不足是可以接受的.

4.2.2 作业集整体完成率的对比实验

在混合关键级作业的调度中,由于调度的目标不仅仅是保证高关键级的作业完成,而也应尽可能多地调度低关键级作业,因此作业集的整体完成率是混合关键级作业调度中的另一个重要指标.如果作业集的整体完成率较高,意味着作业集中更多的作业能够完成,系统的计算资源能得到更好的利用,算法性能也相应地更好.关于作业集整体完成率的对比实验,如图 5 和图 6 所示.

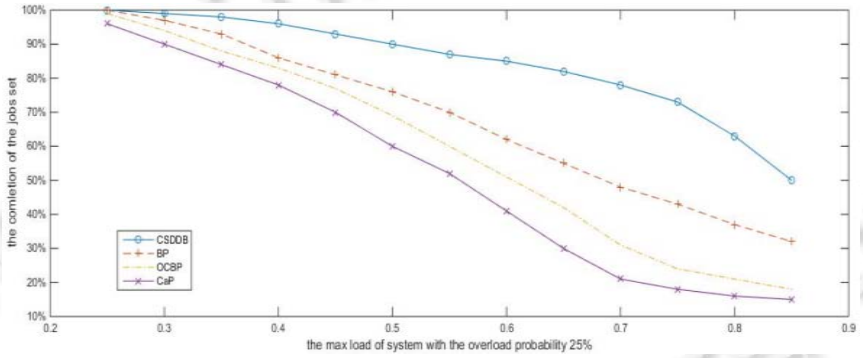

Fig.5 Complement ratio of jobset τ when P_χ=25%

图 5 当 P_χ=25%时作业集 τ的完成率

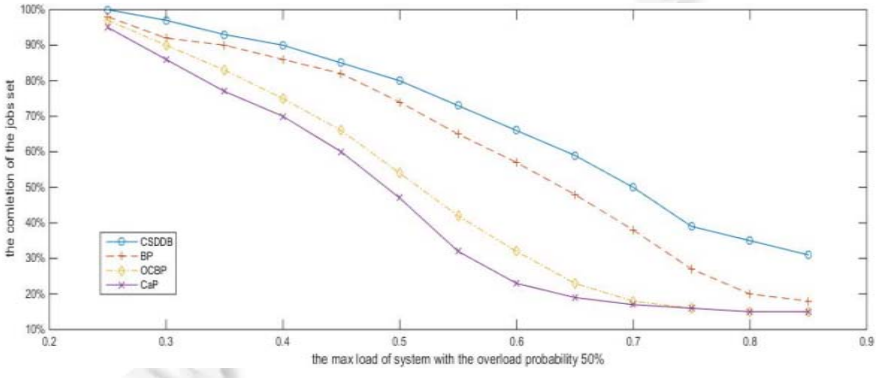

Fig.6 Complement ratio of jobset τ when P_χ=50%

图 6 当 P_χ=50%时作业集 τ的完成率

在图5和图6中,x轴表示系统负载;y轴表示作业集的完成率,为20次运行结果的平均值.可以看出,CSDDB在作业集的完成率方面具有较好的性能.特别是在系统负载较高时,OCBP和CaP的完成率保持在17%左右.这意味着几乎只有最高关键级的作业完成,但CSDDB的完成率仍能保证30%左右的作业能够完成.这是因为CSDDB能够通过松弛时间把关键级切换尽可能地延后,在整个运行过程中,能够满足更多的作业完成.显然,在作业的完成率方面,CSDDB具有明显的优势.

在实际的应用中,作业完成率性能的优势对于一个系统而言至关重要.特别是在当今的嵌入式系统中,正确性不再是系统运行的唯一目标.在很多情况下,低关键级作业的完成能够为系统带来很多的优势,比如使用者的体验性等.因此,在保证系统基本的安全性、可靠性能够得到保证的前提下,尽可能地使得混合关键级作业系统的平均系统关键级降低,同时使更多的作业能够得到完成,是混合关键级系统调度问题的重点之一.CSDDB在混合关键级作业系统的调度上,充分考虑了低关键级作业的执行,通过系统执行关键级的切换,尽可能在保证高关键级作业完成的情况下,为低关键级作业提供更多的执行机会.实验证明,CSDDB具有较好的性能.

5 结束语

本文针对混合关键级作业系统,在传统实时调度理论中的时间需求边界函数的基础上,提出了混合关键级系统中的动态时间需求边界函数.该函数考虑了作业在不同关键级上的执行时间、可能被阻塞的情况以及在实例中作业执行关键级的变化可能带来的影响,以矢量的形式表述了一个混合关键级作业在不同关键级上的时间需求边界,且该边界随着系统运行产生动态的变化.在动态需求边界函数的基础上,本文提出了作业松弛时间的概念,并进一步提出了关键级松弛时间的概念,以之为根本参数,提出了一种基于动态需求边界的混合关键级作业调度算法CSDDB.该算法总是选取不小于0的松弛时间中的最小值所对应的关键级作为系统的执行关键级,通过系统执行关键级的切换,尽可能地在不损害高关键级作业执行的情况下,为低关键级作业创造运行机会,从而提高作业集的整体完成率.仿真实验证明,CSDDB在系统关键级与作业的完成率方面具有较优的性能.

当系统的运行环境为多处理器平台,特别是异构多处理器平台,或者系统的目标涉及功耗、能量等其他因素时,动态需求边界的描述和应用都会变得更复杂,这也是本文继续研究的主要方向.

References:

[1] Vestal S. Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance. In: Proc. of the 28th IEEE Real-time Systems Symp. (RTSS). IEEE, 2007. 239−243.

[2] Baruah S, Bonifaci V. Scheduling real-time mixed-criticality jobs. IEEE Trans. on Computer, 2011,61(8):1140−1152.

[3] Baruah S. Schedulability analysis for a general model of mixed-criticality recurrent real-time tasks. In: Proc. of the IEEE Real-Time Systems Symp. 2016. 25−34.

[4] Burns A, Davis RI. Adaptive mixed criticality scheduling with deferred preemption. In: Proc. of the IEEE Real-time Systems Symp. 2014. 21−30.

[5] de Niz D, Lakshmanan K, Rajkumar R. On the scheduling of mixed-criticality real-time task sets. In: Proc. of the IEEE Real-time Systems Symp. IEEE, 2009. 291−300.

[6] Huang H, Gill C. Implementation and evaluation of mixed-criticality scheduling approaches for period tasks. In: Proc. of the Real Time and Embedded Technology and Application Symp. IEEE, 2012. 23−32.

[7] Lakshmanan K, de Niz D, Rajkumar R. Mixed-criticality task synchronization in zero-slack scheduling. In: Proc. of the 17th Real-Time and Embedded Technology and Applications Symp. (RTAS). IEEE, 2011. 47−56.

[8] Li H, Baruah S. An algorithm for scheduling certifiable mixed-criticality sporadic task systems. In: Proc. of the Real-time Systems Symp. San Diego: IEEE Computer Society Press, 2015. 183−192.

[9] Baruah S, Bonifaci V, D'Angelo G, Marchetti-Spaccamela A, van der Ster S, Stougie L. Mixed-criticality scheduling of sporadic task systems. In: Proc. of the 19th Annual European Symp. on Algorithms. Saarbrucken: Springer-Verlag, 2013. 555−566.

[10] Li H. Scheduling mixed-criticality real-time systems [Ph.D. Thesis]. University of North Carolina at Chapel Hill, 2013.

[11] Ekberg P, Yi W. Bounding and shaping the demand of mixed-criticality sporadic tasks. In: Proc. of the 24th Euromicro Conf. on Real-time Systems (ECRTS). IEEE, 2012. 135−144.

[12] Ren J, Phan LTX. Mixed-criticality scheduling on multiprocessors using task grouping. In: Proc. of the Euromicro Conf. on Real-Time System. 2015. 25−34.

[13] Davis R, Burns A. Robust priority assignment for fixed priority real-time systems. In: Proc. of the IEEE Real-time System Symp. 2007. 3−14.

[14] Huang PC, Kumar P, Stoimenov N, Thiele L. Interference constraint graph—A new specification for mixed-criticality systems. In: Proc. of the IEEE 18th Conf. on Emerging Technologies & Factory Automation. IEEE, 2013. 1−8.

[15] Bate I, Burns A, Davis RI. An enhanced bailout protocol for mixed criticality embedded software. IEEE Trans. on Software Engineering, 2017,43(4):298−320.

[16] Bate I, Burns A, Davis RI. A bailout protocol for mixedcriticality systems. In: Proc. of the 27th Euromicro Conf. on Real-Time System. 2015. 259−268.

[17] Neukirchner M, Lampka K, Quinton S, Ernst R. Multi-mode monitoring for mixed-criticality real-time systems. In: Proc. of the 9th IEEE/ACM/IFIP Int'l Conf. on Hardware/Software Codesign and System Synthesis. IEEE Press, 2013. 34−43.

[18] Gu X, Easwaran A. Dynamic budget management with service guarantees for mixed-criticality systems. In: Proc. of the IEEE Real-Time Systems Symp. 2016. 47−56.

[19] Abdeddaïm Y. Probabilistic schedulability analysis for fixed priority mixed criticality real-time systems. In: Proc. of the Design, Automation and Test in Europe (DATE). 2016. 596−601.

[20] Huang LD, Li RF. Scheduling of mixed criticality real-time tasks set with deadline as the critical parameter. Journal of Computer Research and Development, 2016,53(7):1641−1647 (in Chinese with English abstract).

附中文参考文献:

[20] 黄丽达,李仁发.截止时限为关键参数的混合关键级实时任务调度研究.计算机研究与发展,2016,53(7):1641−1647.

曾理宁(1986－),男,博士,主要研究领域为嵌入式软件,实时调度理论,信息物理融合系统.

杨帆(1985－),男,博士,讲师,CCF 专业会员,主要研究领域为数据建模,嵌入式系统.

徐成(1962－),男,博士,教授,博士生导师,CCF 专业会员,主要研究领域为嵌入式系统,智能机器人.

徐洪智(1974－),男,博士,副教授,主要研究领域为嵌入式系统,并行计算.

李仁发(1956－),男,博士,教授,博士生导师,CCF 杰出会员,主要研究领域为计算机系统结构,嵌入式系统,信息物理融合系统.