软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2014,25(2):284−297 [doi: 10.13328/j.cnki.jos.004534]

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

多处理器混合关键性系统中的划分调度策略*

谷传才, 关 楠, 于金铭, 王 义, 邓庆绪

(东北大学 信息科学与工程学院,辽宁 沈阳 110819)

通讯作者: 邓庆绪, E-mail: dengqx@mail.neu.edu.cn

摘 要: 多核处理器正越发广泛地应用到现代嵌入式系统的设计与实现当中,其强大的计算能力为将多个不同关键性级别的功能子系统集成到统一的共享资源平台提供了支持.混合关键性系统的调度问题即使在单处理器平台中都极具挑战性,在多处理器平台则更为困难.将目前资源利用率最高的单处理器混合关键性调度算法EY-VD扩展到多处理器平台中.首先,结合传统的划分调度策略提出了适用于多处理器混合关键性系统的 MC-PEDF(mixed-criticality partitioned earliest deadline first)划分调度算法.尽管比之前的算法有更好的可调度性能,但传统的划分策略不能有效地平衡不同关键性级别下的负载,故其不完全适用于混合关键性系统.为了克服传统策略的不足,提出了划分调度策略 OCOP(one criticality one partition).OCOP 允许系统在关键性模式切换时对实时任务集进行重新划分,进而更好地平衡各个处理器在不同关键性模式中的资源利用率.基于 OCOP,提出了第 2 种划分调度算法 MC-MP-EDF(mixed-criticality multi-partitioned EDF).基于随机生成任务集的仿真实验结果表明,与 MC-PEDF 和已有的算法相比,MC-MP-EDF 能够显著地提高系统的可调度性,尤其是在处理器数量较多的系统中.

关键词: 混合关键性系统;多处理器;划分调度;EDF(earliest deadline first)

中图法分类号: TP316 **文献标识码**: A

中文引用格式: 谷传才,关楠,于金铭,王义,邓庆绪.多处理器混合关键性系统中的划分调度策略.软件学报,2014,25(2):284−297. http://www.jos.org.cn/1000-9825/4534.htm

英文引用格式: Gu CC, Guan N, Yu JM, Wang Y, Deng QX. Partitioned scheduling policies on multi-processor mixed-criticality systems. Ruan Jian Xue Bao/Journal of Software, 2014,25(2):284−297 (in Chinese). http://www.jos.org.cn/1000-9825/4534.htm

Partitioned Scheduling Policies on Multi-Processor Mixed-Criticality Systems

GU Chuan-Cai, GUAN Nan, YU Jin-Ming, WANG Yi, DENG Qing-Xu

(School of Information Science and Engineering, Northeastern University, Shenyang 110819, China)

Corresponding author: DENG Qing-Xu, E-mail: dengqx@mail.neu.edu.cn

Abstract: Multi-Core processors are more and more widely used in embedded systems as they provide great computing capacities to integrate multiple functionalities with different criticality levels into a shared platform. The scheduling problem of mixed-criticality systems appears to be challenging, even on single-processor platforms. This work extends the state-of-the-art single-processor mixed-criticality scheduling algorithm EY-VD to multi-processor systems. To begin with, it integrates EY-VD into traditional workload partitioning schemes to get a multiprocessor mixed-criticality scheduling algorithm MC-PEDF (mixed-criticality partitioned earliest deadline first). Although MC-PEDF performs better than previous solutions, the study finds that the traditional workload partitioning schemes are not suitable for mixed-criticality systems as it does not explore the asymmetricity of workload on different criticality levels. To overcome this problem, a workload partitioning policy OCOP (one criticality one partition) is proposed. OCOP allows tasks to be reassigned to a different processor when criticality mode switch occurs, thus can better balance the resource utilization among processors on different criticality levels. Based on OCOP, the second partitioned scheduling algorithm MC-MP-EDF (mixed-criticality multi-

* 基金项目: 国家科技支撑计划(2012BAF13B08); 中央高校基本科研业务费项目(FRFCUN100204001, FRFCUN110804003); 国家自然科学基金(61300022)

收稿时间: 2013-05-07; 修改时间: 2013-09-29; 定稿时间: 2013-12-05

partitioned EDF) is constructed. Experiments with randomly generated workload show that MC-MP-EDF can drastically improve the system schedulability comparing with MC-PEDF and other previous algorithms, especially for systems with more processors.
Key words: mixed-criticality system; multi-processor; partitioned scheduling; EDF (earliest deadline first)

随着信息技术日新月异的发展,现代嵌入式实时系统中集成功能的规模和复杂性也呈现爆炸式增长的趋势.例如,当今汽车电子系统中已经集成了数十个甚至上百个微处理器,而航空电子系统中微处理器的数量则更为庞大.为了适应未来日益复杂、庞大的系统功能需求,以及满足嵌入式实时系统硬件本身体积、成本、能耗等诸多方面的约束,将多个在传统设计中部署在独立子系统中的不同关键性功能应用集成到共享处理器资源的单一硬件平台,成为当今嵌入式实时系统设计的趋势和发展方向.而多处理器平台的出现及其迅猛发展,在极大地提升了处理器性能的同时,也为嵌入式系统功能集中化的设计方案提供了硬件性能的保障.然而,与传统的实时调度问题相比,混合关键性系统中的实时调度和可靠性认证等问题更为复杂和困难,而传统的经典调度算法(如 EDF)在混合关键性系统中的资源利用率十分低下,不能直接应用于系统设计.

2007 年,Vestal 在文献[1]中首先形式化定义了混合关键性系统(mixed-criticality system)中的实时调度问题.之后,混合关键性系统的研究迅速成为近年来实时系统领域的热点问题,并引起了大量知名学者的关注.然而,大量的工作都集中于单处理器平台上的混合关键性系统调度问题.近年来,多处理器平台中的混合关键性系统调度问题受到了广泛的关注.

根据是否允许相同任务释放的作业在运行时执行在不同的处理器上,传统的(单关键性)多处理器调度算法通常被分为全局调度和划分调度.文献[2]证明:在强实时系统中,划分调度具有更好的可调度性.Kelly 等人在文献[3]中首次提出了可抢占多处理器平台中基于固定优先级算法的划分调度问题,并分别对比了采用资源利用率降序排列(decreasing utilization,简称 DU)和关键性降序排列(decreasing criticality,简称 DC)这两种实时任务集排序策略以及首次适应降序(first-fit decreasing,简称 FFD)、最差适应降序(worst-fit decreasing,简称 WFD)等启发式划分策略对划分调度性能的影响.Baruah 等人在文献[4]中提出了基于非固定优先级算法 EDF-VD(EDF with virtual deadlines)[5],并采用基于 DC 策略的混合关键性划分调度算法 MC-Partition.

然而据我们所知,目前在单处理器平台下,Ekberg 等人在文献[6]中提出的算法(本文简称 EY-VD)的资源利用率远高于其他已有的混合关键性算法,但仍未有基于该算法的混合关键性划分调度问题的研究.而已有的划分调度算法均基于传统非混合关键性多处理器系统中的划分策略,尚未有特别针对混合关键性系统优化的划分调度策略的研究.因此,本文首先将单处理器平台中的非固定优先级混合关键性调度算法 EY-VD 扩展至多处理器平台,并基于传统划分策略提出了第 1 种划分调度算法 MC-PEDF(mixed-criticality partitioned earliest deadline first).然后,通过对传统划分策略可能导致混合关键性系统在不同关键性模式中处理器间资源利用率不平衡问题的研究,我们提出了针对多处理器平台混合关键性系统优化的划分调度策略 OCOP(one criticality one partition).OCOP 划分策略允许混合关键性实时任务在不同的系统关键性模式下被分配到不同的处理上执行.该方法较好地平衡了系统在不同关键性模式中各个处理器间的资源利用率,进而提升了划分调度算法的性能.基于 OCOP 划分策略,我们提出了第 2 种划分调度算法 MC-MP-EDF(mixed-criticality multi-partitioned EDF).

本文第 1 节具体介绍系统模型和相关工作.第 2 节详细描述基于传统划分策略的 MC-PEDF 算法及其理论分析和证明.第 3 节讨论传统划分策略在多处理器混合关键性系统中的缺陷与局限性,并提出针对混合关键性系统优化的新型划分策略 OCOP 和基于该划分策略的划分调度算法 MC-MP-EDF.第 4 节通过仿真实验对比本文提出的两种算法与之前算法的性能以及 OCOP 划分策略的优化效果.第 5 节对全文做简要的总结.

1 研究基础

1.1 系统模型与定义

在本节,我们对本文所用的混合关键性实时任务系统模型给出形式化的定义,并阐述一些相关的基本术语和概念.与传统的偶发任务模型类似,我们将混合关键性偶发实时任务系统定义为由一组有限数量且相互独立

的混合关键性偶发实时任务构成的集合,其中,每个偶发实时任务都将产生任意可能次序的无限数量混合关键性作业序列.

1.1.1 混合关键性任务和混合关键性作业

我们将每个混合关键性任务定义为一个四元组:$\tau_i=(T_i,D_i,C_i,\zeta_i)$.其中元素的具体含义描述如下:

- $T_i\in R^+$,偶发任务的周期,表示任意两个连续释放作业间的最小时间间隔(不同于周期任务的固定周期).
- $D_i\in R^+$,表示任务的相对截止期.
- $C_i\in N^+\to N^+$,为任务的最差执行时间(WCET)函数,返回任务在某个特定关键性级别下的评估 WCET 取值.例如,$C_i(\ell)$表示任务在关键性级别为ℓ时的 WCET.
- $\zeta_i\in\{1,2,\ldots,L\}$,表示任务的关键性级别,约定该值越大,表示的关键性级别越高.L 为该混合关键性系统中关键性级别的最大值.

需要注意的是,任意任务的相对截止期与周期之间没有任何约束关系,即,相对截止期可以小于、等于甚至大于周期.

混合关键性任务τ_i释放的第 j 个作业记为 J_i^j, 释放时间 $r_i^j\in R^+$, 绝对截止期 $d_i^j=r_i+D_i$, 完成时间 f_i^j, 其中,三者大小关系为 $r_i^j\leqslant f_i^j\leqslant d_i^j$, 另外规定任务$\tau_i$中释放的所有作业都具有相同的 C_i 和关键性级别ζ_i.

为了简化系统的描述和算法分析,本文只研究包含两个关键性(即 L=2)的双关键性模型.特别地,用 LO 表示低关键性级别,用 HI 表示高关键性级别.本文的结论都可以扩展到任意关键性个数的情况.

1.1.2 混合关键性偶发性实时任务系统的运行时行为

混合关键性作业 J_i^j 在时刻 r_i^j 由任务τ_i 释放,需要执行 γ_i^j 单位时间. J_i^j 在被标志完成运行之前, γ_i^j 准确地值无法确定.通过测量作业,运行过程中 γ_i^j 值可以用来判定混合关键性系统的运行时行为.当整个系统满足公式(1)的约束时,称系统的运行时行为是λ关键性级别:

$$\lambda=\max_{\forall\gamma_i^j}\{\ell\,|\,\min\{\ell\,|\,\gamma_i^j\leqslant C_i(\ell)\}\}\tag{1}$$

特别地,任何作业 J_i^j 运行时间超过 $C_i(L)$而未结束执行的运行时行为被定义为错误运行行为.本文接下来的部分,假定系统不会经历错误运行行为.公式(1)暗含每个运行中的作业 J_i^j 的执行时间 γ_i^j 越长,将导致系统经历更高关键级别运行时行为的可能性越大.所有作业中的最高关键性运行时行为决定系统的运行时行为.

1.1.3 混合关键性任务系统的可调度性

混合关键性任务系统被要求对每个关键性级别进行独立的运行时可调度性认证.依据以上原则,本文给出调度算法 A 对于混合关键性任务系统π可调度的定义.

定义 1(混合关键性任务系统的可调度性). 给定一种调度算法 A,混合关键性任务系统可调度的充分必要条件是:当系统经历任意可能的λ关键性运行时行为时,都能够满足公式(2)的约束条件:

$$\forall\tau_i:\zeta_i\geqslant\lambda\Rightarrow\forall J_i^j:J_i^j\text{在截止期}d_i^j\text{之前执行完毕}\tag{2}$$

通过以上定义可知:当系统显示λ关键性级别行为时,所有关键性级别低于λ的实时任务释放的作业直接被放弃执行不需要满足截止期的约束;混合关键性系统的正确性原则仅要求设计者保证系统中所有关键性大于等于当前系统运行时关键性级别λ的作业执行时间不超过其截止期,并不需要保证任务τ_i 释放的作业在系统经历更高关键性行为时满足截止期的约束.

1.1.4 需求上界函数 DBF

需求上界函数(demand-bound function,简称 DBF)[7]描述任意指定长度时间间隔内,系统产生的最大执行时间需求上界,即系统所需提供的能够保证实时性约束的最小处理器资源数量.

定义 2(需求上界函数——DBF). 一个需求上界函数 $dbf(\tau_i,\Delta)$表示一个实时任务τ_i在给定长度为Δ的时间间隔内,可能产生的最大执行时间需求的上界.其中,执行时间的需求为所有由实时任务τ_i 释放,且调度窗口完全包含在长度为Δ的时间间隔内的作业所需要的最大执行时间(即 WCET)之和.

DBF 是传统实时任务模型下分析实时系统工作量可调度性(schedulability)的有效方法,并且针对诸多主流

的实时任务模型,都有精确的计算方法.例如,在偶发实时任务模型中,DBF 可以通过文献[7]中的算法求得.Ekberg 等人在文献[6]中将其应用扩展到单处理器混合关键性偶发实时任务系统模型中,并给出了基于 EY-VD[6]算法的混合关键性实时任务分别在低关键性和高关键性下的 DBF 计算方法.下面我们将对该方法进行简要介绍.

当混合关键性系统运行在低关键性级别($\ell=LO$)时,每个混合关键性任务 τ_i 可视为普通的单关键性实时任务 $(C_i(LO),D_i(LO),T)$.其中,$D_i(LO)$是实时任务 τ_i 在低关键性级别下的相对截止期,当 τ_i 为低关键性实时任务时,$D_i(LO)=D_i$;当 τ_i 为高关键性实时任务时,$D_i(LO)\leqslant D_i(HI)=D_i$.可以得到:

$$dbf_{LO}(\tau_i,t)=\max\left\{0,\left(\left\lfloor\frac{t-D_i(LO)}{T_i}\right\rfloor+1\right)\cdot C_i(LO)\right\} \tag{3}$$

当系统运行时切换到高关键性级别时,这里面可能包含一个遗留作业.而从高关键性任务 τ_i 释放的遗留作业的调度窗口最小可能为 $D_i(HI)-D_i(LO)$.据此,可以得到高关键性任务 DBF 的最大值:

$$full(\tau_i,t)=\max\left\{0,\left(\left\lfloor\frac{t-(D_i(HI)-D_i(LO))}{T_i}\right\rfloor+1\right)\cdot C_i(HI)\right\} \tag{4}$$

函数 $full(\tau_i,t)$的计算过程中没有考虑遗留作业在系统的关键性变化前已经执行的时间.公式(5)用于计算该时间,其中,$n=t \bmod T_i$.

$$done(\tau_i,t)=\begin{cases}\max\{0,C_i(LO)-n+D_i(HI)-D_i(LO)\}, & \text{if } D_i(HI)>n\geqslant D_i(HI)-D_i(LO)\\ 0, & \text{otherwise}\end{cases} \tag{5}$$

根据公式(4)和公式(5),可以得到高关键性实时任务 τ_i 在高关键性模式下的需求上界函数为

$$dbf_{HI}(\tau_i,t)=full(\tau_i,t)-done(\tau_i,t) \tag{6}$$

dbf_{LO} 和 dbf_{HI} 的计算复杂性均是伪多项式的.

1.2 相关工作

在混合关键性调度问题的研究中,很多文献对由有限个混合关键性作业组成之集合的调度问题进行了研究[8,9].Baruah 等人在文献[10]中已证明:用最优的算法判断一个给定的作业集合是否可以被调度,是强 NP 难问题.因此,对于混合关键性任务实时调度的研究重点集中在找到在实践中有很好性能的次最优调度算法.

Baruah 等人在文献[5]中提出一种基于适用于混合关键性系统的改进型 EDF 算法 EDF-VD.该算法提出了一个虚拟截止期(virtual deadline)的概念,每个高关键性任务有两个虚拟相对截止期(均不大于真实的相对截止期),其中一个被视为系统处于低关键性模式时高关键性任务的相对截止期,另一个是系统处于高关键性模式时高关键性任务的相对截止期.当实时任务系统的资源利用率满足一定条件时,可以利用此 EDF 算法调度.Ekberg 等人在文献[6]中提出了用于计算采用虚拟相对截止期的实时任务在不同关键性模式下 DBF 的方法,并基于该方法提出了一种有效的为混合关键性实时任务系统的高关键性任务计算虚拟相对截止期的贪心算法 EY-VD.本文提出的算法在每个处理器上的调度行为与 EY-VD 算法类似,高关键性任务释放的作业在不同的关键性模式下使用不同的相对截止期.另外,本文在进行算法分析时也使用了文献[6]中的一些结论.

针对多处理器平台中混合关键性实时系统的调度问题,Mollison 等人在文献[11]中将不同关键性级别的任务封装在一起,然后在多核系统中用关键性单调优先级分配策略调度这些分组.在文献[12]中,Pathan 将多处理器系统中传统的全局调度算法扩展到混合关键性系统中.在文献[4,13]中,Baruah 和 Li 等人研究了 EDF-VD 算法在多处理器系统中的扩展,并提出了全局调度算法 Global-fpEDF 和划分调度算法 MC-Partition.在文献[3]中,Kelly 等人研究了固定优先级调度算法在多处理器混合关键性系统中的划分调度问题.

另外,De Niz 等人在文献[14]中提出了用于调度偶发性混合关键性任务的零松弛时间算法.这种算法的关键点在于,通过动态地调整作业优先级来避免低关键性作业对高关键性作业造成的运行时调度干扰.这种方法已经被扩展到用于不可抢占共享资源平台[15]和分布式平台.在这些平台上,混合关键性任务需要被分配到不同的执行单元[16].在文献[17]中,Li 等人将 OCBP(own criticality based priority)算法[8]扩展到偶发实时任务模型,提

出了伪多项式复杂度的在线算法.

2 基于传统划分策略的混合关键性划分实时调度算法

在众多的单处理器混合关键性实时调度算法中,EY-VD 算法[6]拥有比 RMS(rate-monotonic scheduling)[19], OPA(optimal priority assignment)[20],EDF-VD[5]等其他已知的混合关键性调度算法更高的资源利用率.现有多处理器平台下的混合关键性实时调度算法的研究还十分有限.本文研究 EY-VD 算法在混合关键性多处理器平台上的扩展,并提出了一种基于划分调度的算法.

2.1 多处理器划分调度的基本方法

传统的隐式截止期(实时任务的相对截止期严格等于周期)周期性实时任务集在可抢占多处理器平台中的划分调度问题是强 NP 难的(经典的强 NP-难问题 Bin-Packing[21]可以转换成该问题),而该问题的计算复杂性很容易扩展到更加一般化的混合关键性偶发实时任务模型,因此,多处理器平台中混合关键性偶发实时任务系统的划分调度问题也是强 NP 难的.

在传统(非混合关键性)实时任务模型下,关于可抢占多处理器平台中的划分调度问题已经存在广泛的研究成果,而这些成果所采用的启发式划分策略大多可以分解为如下 3 个典型的步骤:

(1) 实时任务排序.通常,在将实时任务分配到特定处理器之前都会遵从某种策略将待分配的实时任务集排序,再依次进行分配处理器的操作.

(2) 处理器选择.按照第 1 步的排序结果,依次为实时任务集中的任务使用某种启发式策略分配处理器.

(3) 划分检验.检验分配到某处理器中的实时任务子集能否被某种单处理上的实时调度算法所调度.

对于划分检验,通常是采用单处理器下实时调度算法的可调度性测试方法对分配到该处理器上的实时任务子集进行可调度性分析.在此步骤中,不需要考虑多处理器的资源分配问题,而且存在大量单处理器上十分成熟的调度算法及其可调度性测试方法可以选择,因此,传统的多处理器实时任务划分调度问题的研究很多都集中在实时任务的排序及划分处理器的选择策略上.

处理器的选择策略和 Bin-Packing 问题的启发式策略十分类似,主要包括:

- 首次适应(first fit,简称 FF).每次为实时任务选处理器时,按照固定的处理器顺序选择第 1 个满足划分检测条件的处理器,并将该实时任务分配到此处理器中.
- 最好适应(best fit,简称 BF).每次将实时任务分配到满足划分检测条件,且剩余资源最少的处理器中.
- 最坏适应(worst fit,简称 WF).与 BF 相反,WF 每次选择满足划分检测条件,但剩余资源最多的处理器.

为了使启发式分配策略达到更好的可划分性能,在实时任务选择处理器划分之前,采用合适的策略对实时任务集进行预排序是必要的.

特别地,文献[3,4]中提出的各种混合关键性实时任务系统的划分调度算法也基本上遵从上述的原则.

2.2 MC-PEDF算法描述

本节介绍本文提出的可抢占多处理器平台中,基于可调虚拟截止期 EY-VD 算法且采用传统划分策略的混合关键性偶发实时任务系统划分调度算法 MC-PEDF.

假设一个混合关键性偶发实时任务实时系统包含一个混合关键性实时任务集合π(由 n 个相互独立的偶发性实时任务$\tau_1,\tau_2,\ldots,\tau_n$组成)和一个多处理器系统$\Pi$(包含 m 个单位速率的同质处理器 $p_1,p_2,\ldots,p_m$).MC-PEDF 是 Bin-Packing 问题中 FF-Decreasing 启发式策略的一个变种(经过仿真实验的分析我们发现:在采用 EY-VD 作为划分处理器调度算法时,FF 启发式策略要优于 BF 以及 WF 等启发式策略.但对比这些启发式策略不是本文的重点,故在此不做详细的分析),该算法的伪代码描述如图 1 所示.

在算法的第 1 行,将混合关键性实时任务集π按照关键性降序排序,而对于相同关键性的任务按平均利用率降序排列.算法的第 3 行~第 10 行将按照第 1 行的排序结果依次为每个实时任务分配处理器;在分配处理器时,MC-PEDF 会按第 5 行~第 9 行的方法,找到满足划分检测条件并且指标值最小的处理器,并将当前任务分配给

它(第 7 行).

如果在某一次的迭代中,所有的处理器都不能满足分配当前任务的划分检测条件,则算法会在第 3 行因 t 等于 *None* 而退出循环,并在第 12 行返回“FAILURE”.

如果对于所有的实时任务都能被正确的分配处理器,则算法会在第 3 行因待分配实时任务集合π为空集而退出循环,最后在第 13 行返回“SUCCESS”.

算法. MC-PEDF,基于传统划分策略的划分调度算法.
输入:被划分任务集π:={$\tau_1,\tau_2,\ldots,\tau_n$}.
输出: 划分失败返回 FAILURE.
划分成功返回 SUCCESS,以及划分结果Π:={p_1:={·},p_2:={·},…,p_m:={·}}

```
1.    Sort π with Criticality Deceasing order
2.    t:=None
3.    while t=None and π≠∅ do
4.      t:=PopFisrtElement(π)
5.      for each p_i in Π by indicator increasing order do
6.        if CheckSchedulablityWithDBF(p_i∪{t}) then   //基于公式(3)、公式(6)
7.          p_i:=p_i∪{t}
8.          t:=None; exit for
9.        end if
10.     end for
11.   end while
12.   if t≠None return FAILURE   //存在没有被划分的任务
13.   return SUCCESS
```

Fig.1 Pseudo code of MC-PEDF algorithm

图 1 MC-PEDF 算法的伪代码

MC-PEDF 第 6 行的 *CheckSchedulablityWithDBF* 函数基于 Ekberg 等人提出的基于虚拟截止期的 DBF-LO 和 DBF-HI 方法(如公式(3)、公式(6)所示)来检测划分实时任务子集的可调度性.该方法也是目前在混合关键性实时任务集可调度比率评价标准下,可抢占单处理器平台中混合关键性偶发实时任务调度性能最好的算法.

MC-PEDF 的运行时调度算法如下:

如果 MC-PEDF 算法返回 SUCCESS,则系统遵守如下的原则约束进行运行时调度:

(1) 系统启动时,将所有实时任务按 MC-PEDF 的划分结果分配到各处理器中,且系统进入低关键性模式.

(2) 对于每个处理器,均保证在任意时刻占用处理器执行的作业之优先级不低于其就绪队列中任何其他作业的优先级(优先级的取值越小,所表示的优先级越高).即,任何时刻处理器只允许分配到其中的任务所释放的优先级最高的未完成作业占用处理器执行,当更高优先级作业到来时,会抢占当前执行的作业.

(3) 当系统运行在低关键性模式时,所有低关键性任务和高关键性任务释放的作业均采用其释放时间与低关键性下的虚拟相对截止期(注意,本文约定低关键性任务的虚拟相对截止期等于其低关键性下相对截止期)之和作为其运行时优先级,并被插入其所分配处理器的就绪队列等待调度执行.

(4) 当系统因正在执行作业的执行时间超过低关键性下的最差执行时间而未发出执行完成信号时,系统将进行模式切换,并进入高关键性模式.在模式切换时,所有低关键性任务释放的作业将被终止执行并被清除出就绪队列,而高关键性任务释放作业的优先级将被调整为该作业的释放时间与高关键性下相对截止期之和,然后,系统按照第 2 条约束的原则继续运行时调度.

(5) 当系统运行在高关键性模式时,所有低关键性任务释放的作业将不被添加到就绪队列而是直接终止执行;而高关键性任务释放的作业将采用其释放时间与高关键性下的相对截止期之和作为其运行时优先级,并被插入其所分配处理器的就绪队列等待调度执行.

2.3 MC-PEDF划分算法的时间复杂性和正确性分析

对于算法的时间复杂度,易知:第 1 行的排序算法复杂度不会超过 $O(n^2)$,其中,n 为实时任务集中的任务数量.另外在第 6 行,分配到处理器 p_i 上的实时任务子集(其包含任务的数量小于等于 n)的可调度性判定和第 12 行对各个划分处理器内已分配的实时任务子集中的高关键性任务设置低关键性下的虚拟相对截止期的操作均为伪多项式时间复杂度(文献[6]中的结论).由于第 12 行 for each 循环的次数为划分处理器的个数 m,故第 12 行的时间复杂度亦为伪多项式级别.现在我们分析第 3 行~第 10 行的嵌套循环,其中,外层循环次数不超过实时任务集的任务数量 n,内层 for each 循环次数不超过处理器的数量 m.结合对第 6 行操作的分析,该嵌套循环的时间复杂度为伪多项式级别.综上所述,MC-PEDF 算法的时间复杂度为伪多项式级别.

下面我们讨论 MC-PEDF 算法的正确性.

定理 1. 假设任意的混合关键性偶发实时任务集$\pi\{\tau_1,\tau_2,\ldots,\tau_n\}$和单位速率同质多处理器系统$\Pi\{p_1,p_2,\ldots,p_m\}$.如果 MC-PEDF 算法返回 SUCCESS,且该混合关键性系统采用 MC-PEDF 的运行时调度算法,则该混合关键性系统在低关键性和高关键性模式下都是可调度的.

证明:我们采用反证法证明该定理.如果 MC-PEDF 算法返回 SUCCESS,则所有实时任务一定都分配到了某个特定处理器中.我们假设存在π和Π在算法返回 SUCCESS 时,该混合关键性系统不可调度.不失一般性地,一定存在处理器 p_k,满足分配在 p_k 中的非空实时任务子集在运行时不可调度.

令在 MC-PEDF 算法划分过程中最后一个分配到 p_k 中的任务为 τ_e,则一定有 *CheckSchedulablityWithDBF* 返回真,即,分配到 p_k 中的实时任务子集在低关键性和高关键性下都是运行时可调度的.这与假设相矛盾,也证明了该定理的正确性. □

3 针对混合关键性系统的多次划分实时调度策略

上一节,我们基于传统的划分调度原则提出了 MC-PEDF 算法.本节将分析多处理器平台中混合关键性实时任务运行时的一些特性,并以此来改进传统的划分策略,提出一种针对混合关键性系统更有效的多处理器划分策略 OCOP.最后,提出一种基于 OCOP 的改进型 MC-PEDF 算法 MC-MP-EDF.

3.1 传统划分策略的局限性

在传统的多处理器偶发实时任务模型中,因为只有一种运行时模式,所有任务的主要调度参数(最差执行时间、截止期等)都是恒定的,因此,传统的划分算法可以通过优化的排序和分配策略来平衡各个处理器上分配的负载,以达到较高的资源利用效率.

与传统的多处理器实时调度模型不同,混合关键性系统在运行时会因为运行时系统关键性级别的不同而处于不同的关键性模式.而在不同的模式下,系统中需要调度的实时任务规模和特征参数都有差异.例如,当系统运行在低关键性模式时,所有低关键性任务和高关键性任务释放的作业都存在于就绪队列中,并按照各自的优先级等待调度;而当系统切换到高关键性级别时,系统需要保障高关键性任务释放的作业在更悲观的调度参数(高关键性下更长的评估最差执行时间等)下依然满足截止期约束,故,强行终止所有低关键性任务释放的作业并将其从就绪队列中移除.这种差异性导致某些处理器在不同运行时模式时的资源利用率相差较大.

正因多处理器混合关键性实时任务系统中实时任务调度参数的可变性,使得传统的划分策略很难平衡各个关键性模式中的资源利用率.这使得在划分实时任务时,待划分处理器尽管已分配的实时任务子集在大多数模式下的资源利用率很低,却因为在个别模式下的利用率过高而不能接受新的任务,从而导致资源利用率降低,甚至是实时任务集的不可调度,最终降低了总体的系统资源利用效率.

例 1:考虑将表 1 所示的实时任务集分配到 2 个单位速率处理器 P_1 和 P_2 的情况.容易验证,无论采用 DU 还是 DC 的预排序策略,排序结果都和表中的指标顺序一致;而且容易验证,按照 FFD 的启发式划分策略,实时任务 τ_1 和 τ_2 将被分配到处理器 p_1,而实时任务 τ_3,τ_4 和 τ_5 将被分配到处理器 p_2,且 p_1,p_2 都满足可调度性判定条件.当分配实时任务 τ_6 时,处理器 P_1 上的低关键性模式的资源利用率为 80%,高关键性模式下的资源利用率为 100%.如

果将τ_6分配到p_1,虽然低关键性下的资源利用率增至 92.5%仍小于 100%,但是在高关键性模式下不能通过*CheckSchedulablityWithDBF*的可调度性检验,故τ_6不能被分配到p_1.而p_2上低关键性模式下的资源利用率已经为 100%,故τ_6也不能被分配到处理器p_2上.因此,我们得出表 1 所示的混合关键性实时任务集在 2 个单位速率处理器上不可被 MC-PEDF 划分的结论.但是由于p_2上没有被分配任何高关键性任务,因此在系统进入高关键性模式时,该处理器上的资源利用率为 0.这就造成了p_2在不同关键性模式下处理器资源利用率的极度不平衡,同时造成高关键性模式下处理器p_1和p_2之间的资源利用率相差极大.

Table 1 An example of dual-criticality task set

表 1 双关键性任务集实例

实时任务	$C(LO)$	$C(HI)$	关键性	周期/截止期
τ_1	4	5	*HI*	10
τ_2	4	5	*HI*	10
τ_3	1	1	*LO*	3
τ_4	1	1	*LO*	3
τ_5	1	1	*LO*	3
τ_6	0.5	0.5	*LO*	4

综上所述,传统的划分策略难以平衡混合关键性系统在各个关键性模式下的资源利用率,从而限制了划分调度算法在混合关键性系统中性能的提升空间.因此,如何充分利用高关键性模式下某些处理器中的空闲资源来满足高关键性实时任务的可调度性,并保障更多的低关键性实时任务在低关键性模式下可调度,从而平衡划分处理器在各个关键性模式下的资源利用率,成为本文改进混合关键性实时任务划分策略的主要目标.

3.2 混合关键性模型中的新型划分策略OCOP

观察 1:通过对例 1 的进一步分析我们发现:如果将τ_6分配到p_1,则容易验证p_1在低关键性下是可调度的;而当系统切换到高关键性模式时,由于p_2上没有被分配高关键性的实时任务,因此p_2会一直处于空闲状态,造成资源的极大浪费;而如果在系统关键性模式切换时将p_1中的一个高关键性任务迁移到p_2中继续执行,则很容易验证高关键实时任务τ_1和τ_2在各自独立的处理器中都可以在高关键性模式下满足截止期约束.由此可见,如果允许系统在关键性模式切换时将高关键性任务重新划分,将有助于提高多处理器平台的资源利用率.

基于前文对传统划分策略的分析和观察 1 的启发,我们针对混合关键性实时任务模型提出了新的多处理器划分策略 OCOP.其核心思想是:允许混合关键性实时任务在不同的系统关键性模式下被划分到不同的处理上执行,而系统运行在特定关键性级别时,仍要求每个实时任务释放的作业必须在同一处理器中执行.由于放松了传统划分策略下实时任务释放的作业不能在处理器间迁移的约束,混合关键性系统在模式切换时可以通过更加灵活的模式切换策略来保证更高关键性模式下的可调度性.

对于任意混合关键性实时任务集合π,系统在不同运行时关键性模式下需要保证截止期约束的实时任务是不同的.在低关键性模式下,系统必须保证所有低关键性和高关键性实时任务(即π中的所有任务)的可调度性.我们将所有这些实时任务构成的集合称为低关键性模式调度任务集.类似地,系统在高关键性模式下只需要保证所有高关键性实时任务的可调度性,我们将这些高关键性实时任务构成的集合称为高关键性模式调度任务集.

本文将混合关键性系统的运行时调度分为两种时期:1) 普通时期,即,系统持续运行于某个特定关键性模式的一段时间间隔;2) 模式切换时期,即,从系统监测到当前关键性模式下的过度执行开始,到完成模式切换进入到下一个关键性模式的调度为止的一段连续时间间隔.

OCOP 划分策略对混合关键性多处理器系统的运行时调度存在如下的约束条件:

(1) 规定系统刚启动时处于模式切换时期,完成在低关键性模式下运行的准备工作.

(2) 系统在模式切换时期,允许对实时任务作业进行处理器间迁移操作,将下一个关键性模式调度任务集中的实时任务迁移至下一个关键性模式下的目标处理器中,并终止全部非下一个关键性模式调度任务集中的实时任务作业的执行.

(3) 系统在普通时期不允许任何实时任务作业的处理器间迁移操作,当前关键性模式调度任务集中的实时任务只能将其作业释放到该模式下分配的目标处理器中调度执行.

我们基于双关键性多处理器系统模型对 OCOP 划分策略的主要思想给出如下具体的非形式化描述,而OCOP 也很容易扩展至多关键性多处理器系统模型中:

(1) 低(高)关键性模式调度任务集排序.排序的目的与算法与传统划分策略中的排序类似,区别在于OCOP 需要针对不同关键性模式调度任务集使用不同的算法进行排序,以优化不同关键性模式下划分的性能.

(2) 低(高)关键性模式调度任务集的划分.OCOP 使用与传统划分类似的启发式策略对不同关键性模式的调度任务集分别进行划分,不同模式中调度任务集的划分算法可以不同.特别地,由于在高关键模式中存在低关键性模式中没有执行完的高关键性遗留作业,因此,对高关键性模式调度任务集的划分策略比低关键性模式调度任务集的划分策略难度更高.

(3) 低(高)关键性模式的划分检验.OCOP 同样需要对不同模式中的划分任务进行不同关键性级别的可调度性检验.特别地,由于高关键性遗留作业的影响,高关键性模式划分检验也颇具挑战性.

(4) 划分失败的调整策略.对于传统的划分策略,划分失败直接导致算法结束.OCOP 在某个关键性模式下划分失败时会尝试对实时任务集的一些调度参数进行调整,并重新对各个关键性模式调度任务集进行划分,直至所有关键性模式调度任务集都被正确划分(划分成功),或调度参数的调整空间耗尽(划分失败).

在 OCOP 划分策略下,低关键性模式调度任务集与高关键性模式调度任务集的划分是在两个独立的阶段进行的,这就降低了混合关键性任务在不同运行时关键性模式下调度参数的改变对划分检验造成的干扰,并且能够平衡各个划分处理在不同关键性模式下的资源率,从而提高整个系统的资源利用率和划分性能.在每个关键性模式下,划分失败时的实时任务调度参数调整策略又降低了初始调度参数选择不当对划分性能的影响.

3.3 MC-MP-EDF算法描述

本节我们基于前文提出的OCOP策略,提出新的多处理器平台混合关键性系统划分调度算法MC-MP-EDF.该算法在双关键性模型中的算法描述如图 2 所示,算法中 *CheckSchedulable* 子函数的伪代码描述如图 3 所示.

算法. MC-MP-EDF,基于 OCOP 划分策略的多次划分调度算法.
输入:被划分任务集π:={$\tau_1,\tau_2,\ldots,\tau_n$}.
输出:划分失败返回 FAILURE;划分成功返回 SUCCESS,以及低关键性模式和高关键性模式下的划分结果Π_{LO}:={p_1:={·},p_2:={·},…,p_m:={·}}和Π_{HI}:={p_1:={·},p_2:={·},…,p_m:={·}}.

```
1.  π_HI:=HS:={i|τ_i in π and ζ_i=HI}                    //HS 为可调截止期候选任务集合
2.  for each τ_i in π do set D_i(LO):=D_i−(C_i(HI)−C_i(LO))   //设置 D(LO)为可能的最小值
3.  ct:=None                                              //ct 为上一次调整截止期的任务
4.  while True do
5.    if not CheckSchedulable(π,Π_LO,LO) then             //CheckSchedulable 如图 3 所示
6.      if ct=None then return FAILURE                    //当没有可以回溯的任务时,算法中止
7.      D_ct(LO):=D_ct(LO)+1                              //取消虚拟截止期的最后一次调整
8.      HS:=HS\{ct}                                       //将该任务从候选集合中删除
9.      ct:=None
10.   else
11.     if CheckSchedulable(π_HI,Π_HI,HI) then return SUCCESS
12.     if HS={·} then return FAILURE                     //当没有候选任务时,算法中止
13.     ct:=select a candidate task from HS               //从候选集合中选择调整任务
14.     D_ct(LO):=D_ct(LO)−1
15.     if D_ct(LO)=C_ct(LO) then HS:=HS\{ct}             //虚拟截止期的最小值为 C(LO)
16.   end if
17. end while
```

Fig.2 Pseudo code of MC-MP-EDF algorithm
图 2 MC-MP-EDF 算法的伪代码

算法. *CheckSchedulable*,判断输入的实时任务集是否可调度.
输入:被划分任务集π:={$\tau_1,\tau_2,\ldots,\tau_n$},可用处理器集合 *PS*(保存划分结果),任务的关键性级别 *CL*.
输出:划分失败返回 FAILURE;划分成功返回 SUCCESS,以及划分结果 *PS*.

```
1.    Sort π by decreasing order of C_i(CL)/D_i(CL)
2.    for each p_i in PS do set p_i:={·}
3.    for each itr in TS do
4.      for each p_i in PS by indicator increasing order do
5.        if CheckSchedulablityWithDBF(p_i∪{τ_itr},CL) then      //基于公式(3)、公式(6)
6.          p_i:=p_i∪{τ_itr};
7.          itr:=None; exit for
8.        end if
9.      end for
10.     if itr≠None return False
11.   end for
12.   return True
```

Fig.3 Pseudo code of *CheckSchedulable* algorithm

图 3 *CheckSchedulable* 算法的伪代码

运行时调度算法:

如果 MC-MP-EDF 算法返回 SUCCESS,则系统遵从如下的约束进行运行时调度:

(1) 系统启动时,将所有实时任务按照 MC-MP-EDF 的低关键性划分结果Π_{LO}分配到各处理器中,且系统进入低关键性模式.

(2) 每个处理器均保证:任意时刻,占用处理器执行的作业的优先级不低于其就绪队列中任何作业的优先级.

(3) 当系统运行在低关键性模式时,所有低关键性任务和高关键性任务释放的作业均采用其释放时间与低关键性下的虚拟相对截止期之和作为其运行时优先级,并被插入其低关键性下所分配处理器的就绪队列中等待调度执行.

(4) 当系统发生低关键性模式向高关键性模式切换时,所有低关键性任务释放的作业将被终止执行并被清除出就绪队列,高关键性任务释放作业的优先级将被调整为该作业的释放时间与高关键性下相对截止期之和,并将该作业插入其高关键性下所分配的处理器的就绪队列,同时,从原有就绪队列中移除该作业(对于在两种关键性下分配相同处理器的作业不进行此操作).当所有高关键性任务释放的未完成作业都完成就绪队列迁移之后,系统继续按照第 2 条约束的原则进行调度.

当系统运行在高关键性模式时,所有低关键性任务释放的作业将不被添加到就绪队列而是直接终止执行;而高关键性任务释放的作业将采用其释放时间与高关键性下的相对截止期之和作为其运行时优先级,并被插入其在高关键性下所分配处理器的就绪队列等待调度执行.

3.4 算法正确性分析

引理 1. 假设任意的混合关键性偶发实时任务集π={$\tau_1,\tau_2,\ldots,\tau_n$}和单位速率同质多处理器系统$\Pi${$p_1,p_2,\ldots,p_m$}.如果 MC-MP-EDF 算法返回 SUCCESS,且该任务集采用 MC-MP-EDF 的运行时调度算法,则任意被分配到特定处理器的实时任务释放的作业在低关键性模式下都是可调度的.

证明:由 MC-MP-EDF 的划分算法流程可知,该算法只在第 11 行才会返回 SUCCESS.而在每次外层 for each 循环的迭代过程中,只有在第 5 行的判定条件为假时才可能执行到第 11 行,即,算法返回 SUCCESS 的前提条件是:在所有实时任务使用当前的低关键性虚拟截止期取值时,可以通过第 5 行的低关键性 DBF 判定条件.而第 5 行的判定条件保证所有高关键性和低关键性的实时任务释放的作业,都可以在虚拟截止期之前执行完低关键性下评估的最差执行时间.

又因为对于所有的低关键性实时任务,其虚拟截止期等于低关键性下的截止期,而所有高关键性任务的虚拟截止期又小于等于其低关键性截止期,因此,对于实时任务集里的所有实时任务所释放的作业,在低关键性模

式下都可以保证在其真实截止期之前执行完毕.引理证毕. □

引理 2. 假设任意的混合关键性偶发实时任务集$\pi\{\tau_1,\tau_2,\ldots,\tau_n\}$和单位速率同质多处理器系统$\Pi\{p_1,p_2,\ldots,p_m\}$.如果 MC-MP-EDF 算法返回 SUCCESS,且该任务集采用采用 MC-MP-EDF 的运行时调度算法,则任意高关键性实时任务释放的作业在高关键性模式下是可调度的.

证明:与引理 1 的证明类似,当算法返回 SUCCESS 时,第 11 行的判定条件可以保证在所有实时任务使用当前的低关键性虚拟截止期取值时,系统中所有的高关键性实时任务都可被划分到某个处理器中,并且划分到同一处理器中的高关键实时任务满足 DBF-HI 的判定条件.虽然在系统切换到高关键性模式时,高关键性的实时任务会进行处理器间迁移,但第 5 行的判定条件保障了所有的高关键性实时任务释放的作业在其虚拟截止期之前可以执行完低关键性下的最差执行时间,因此,迁移操作不会影响 DBF-HI 判定的准确性.而 DBF-HI 判定条件也可以保证每个划分处理器上分配的高关键性实时任务所释放的作业都能满足各自高关键性下的截止期约束.引理证毕. □

定理 2. 在 m 个单位速率同质处理器系统中,任何可以被 MC-MP-EDF 算法正确划分的混合关键性偶发实时任务集,在 MC-MP-EDF 的运行时调度算法下均满足在低关键性模式和高关键性模式下都是可调度的.

证明:由算法描述易知,仅当所有的实时任务都能在低关键性下分配到特定的处理中,同时所有的高关键性任务在高关键性下也都能被分配成功的情况下,算法才可能返回 SUCCESS.再综合引理 1 和引理 2 的结论,可以证明定理 2 的正确性. □

3.5 算法复杂性分析

通过 MC-MP-EDF 的算法描述易知,外层 for each 循环进行下一次迭代的必要条件是在第 12 行集合 HS 不为空.又因为每次迭代都会使得 HS 中某个高关键性的实时任务τ_i 的 $D_i(LO)$减 1 或者直接退出循环算法终止,而高关键性任务τ_i 的 $D_i(LO)$减至 $C_i(LO)$时,该任务将从 HS 中移除,因此,外层 for each 循环的最大迭代次数为

$$\sum_{\tau_i\in HI(\tau)}(D_i-C_i(LO)+1) \tag{7}$$

通过与算法 MC-PEDF 类似的复杂性分析易知,第 5 行和第 11 行操作具有伪多项式的时间复杂性.综上分析,MC-MP-EDF 的总体时间复杂性是伪多项式的.

4 实验仿真与结果分析

本节将通过仿真实验来比较本文提出的两种多处理器平台中混合关键性实时任务系统的划分调度算法,以及现有划分调度算法的可调度性能.实验针对双关键性隐式截止期偶发实时任务模型和单位速率的同质多处理器平台.本文的实验实现了 Kelly 等人在文献[2]中提出的两个资源利用率比较高的算法 DC-RM 和 DC-Audsley,并以此为参照,比较本文提出 MC-PEDF 在随机任务集可调度比率评价标准中的性能;并通过观察采用 OCOP 划分策略的算法 MC-MP-EDF 的性能改进效果来评价 OCOP 的有效性.

4.1 随机任务集生成算法

我们采用与文献[6]中类似的生成混合关键性随机任务集的方法,并将其扩展到多处理器平台中.一个随机实时任务集初始时被设置为空集$\pi=\varnothing$,然后逐次添加新的随机混合关键性实时任务.随机任务的生成主要受 5 个参数的控制:随机任务为高关键性任务的最大概率 P_{HI}、高关键性任务的高关键性下的最差执行时间与低关键性下最差执行时间的最大比例 R_{HI}、所有任务在低关键性下最差执行时间的最大值 $C_{LO}^{\max}$、最大的实时任务周期 $T^{\max}$ 和单位速率处理器的个数 m.每个新的随机实时任务按照如下步骤生成:

1. ζ_i服从 P_{HI}的概率取值 HI,否则取值 LO.
2. $C_i(LO)$的取值在 $\{1,2,\ldots,C_{LO}^{\max}\}$范围内服从均匀分布.
3. 如果该随机任务为低关键性任务,则 $C_i(HI)=C_i(LO)$;否则,$C_i(HI)$的取值在$\{C_i(LO),C_i(LO)+1,\ldots,R_{HI}\cdot C_i(LO)\}$范围内均匀分布.

4. T_i的取值在$\{C_i(\zeta_i),C_i(\zeta_i)+1,\ldots,T^{\max}\}$范围内服从均匀分布.

5. 由于我们采用隐式截止期的模型,所以有$D_i=T_i$.

我们定义一个双关键性实时任务集π的平均利用率为

$$U_{avg}(\pi)=\frac{U_{LO}(\pi)+U_{HI}(\pi)}{2} \tag{8}$$

每个任务集在生成时都有一个平均资源利用率基准 U^*.由于产生拥有准确资源利用率的随机任务集比较困难,所以我们允许任务集的平均利用率有一个可接受的误差范围:$U^*_{\min}=U^*-0.005,U^*_{\max}=U^*+0.005$. 若满足$U_{avg}(\pi)<U^*_{\min}$, 就继续产生更多的任务,并将它们添加到任务集$\pi$;如果将一个任务加入$\pi$后导致$U_{avg}(\pi)>U^*_{\max}$, 则随机任务集生成算法将整个任务集抛弃,并从一个空的任务集重新开始生成;如果将一个随机任务加入π后,满足$U^*_{\min}\leqslant U_{avg}(\pi)\leqslant U^*_{\max}$, 则一个随机任务集生成完毕,除非任务集中的所有任务都有相同的关键性或者$U_{LO}(\pi)>0.99\times m$ 或 $U_{HI}(\pi)>0.99\times m$.在这种情况下,此任务集也将被抛弃.在每次实验中,生成随机任务集的各个参数设置分别为:$P_{HI}=0.5,R_{HI}=3,C^{\max}_{LO}=10,T^{\max}=100,m=4$或8.

4.2 实验结果分析

图 4 和图 5 分别比较了 4 个处理器(m=4)和 8 个处理(m=8)平台下,已有划分调度算法 DC-RM[3],DC-Audsley[3],MC-Partition[4]与本文提出的 MC-PEDF,MC-MP-EDF 的随机任务集可调度比率.图中各点数据均基于至少 1 000 个随机任务集样本(图 4 中各点随机任务集平均任务数量为 16~31,图 5 中为 31~63).其中,x 轴是随机任务集的规范化平均资源利用率,y 轴表示可被调度任务集数量与总样本数量的百分比.例如,图 4 中的曲线 MC-MP-EDF 上的点(0.80625,85)表示在多于 1 000 组规范化平均资源利用率在(0.80125,0.81125)范围内的随机任务集样本中,可被 MC-MP-EDF 算法调度的样本数量占总体样本数量的百分比为 85%.

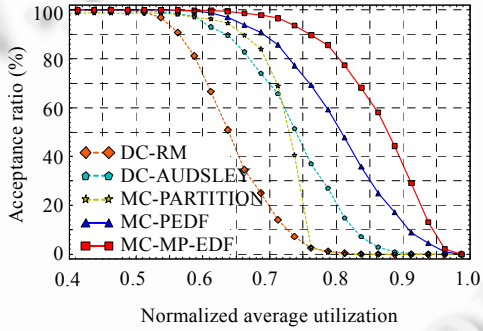


Fig.4 Acceptance comparison on 4-processor platform

图 4 4 个处理器系统中的接受率比较

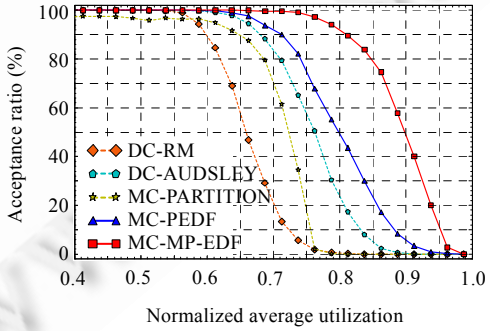


Fig.5 Acceptance comparison on 8-processor platform

图 5 8 个处理器系统中的接受率比较

实验结果表明:本文提出的基于传统划分策略和EY-VD算法的混合关键性多处理划分调度算法MC-PEDF的接受率性能要明显优于之前的算法;而使用针对混合关键性系统的 OCOP 划分策略的改进性划分调度算法MC-MP-EDF的性能也明显优于MC-PEDF算法.从图4与图5的对比中我们可以发现:当混合关键性系统中处理器的数量从4增至8时,MC-MP-EDF相对于其他调度算法的优势更为突出,这也证明了OCOP划分策略对多处理器平台中混合关键性系统的有效性和实用性.

5 结束语

本文在基于传统多处理器划分调度策略实时调度算法的基础上,结合单处理器混合关键性系统中的 EY-VD 算法,提出了多处理器混合关键性系统中基于该算法的划分调度算法 MC-PEDF,并进一步研究了混合关键性系统中划分调度的特殊性质,并在分析传统划分策略局限性的基础上,提出了针对多处理器平台的混合关键性系统划分策略OCOP.与传统的多处理器划分策略相比,OCOP能够有效利用混合关键性实时任务的运行时特征,更好地平衡不同处理器之间在各个运行时关键性模式中的资源利用率,以更为充分、有效地利用系统资源.仿真实验结果表明:MC-PEDF 与先前的多处理器混合关键性实时调度算法相比,具有更高的可调度接受率;基于 OCOP 划分策略的新算法 MC-MP-EDF 又显著提升了 MC-PEDF 的性能;并且系统中处理器的数量越多,优化效果越明显.我们有理由相信,OCOP将会成为多处理器混合关键性划分实时调度的重要分支.

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. [doi: 10.1109/RTSS.2007.47]

[2] Bastoni A, Brandenburg BB, Anderson JH. An empirical comparison of global, partitioned, and clustered multiprocessor EDF Schedulers. In: Proc. of the 31st IEEE Real-Time Systems Symp. (RTSS). IEEE, 2010. 14−24. [doi: 10.1109/RTSS.2010.23]

[3] Kelly OR, Aydin H, Zhao B. On partitioned scheduling of fixed-priority mixed-criticality task sets. In: Proc. of the 10th Int'l Conf. on Trust, Security and Privacy in Computing and Communications (TrustCom). IEEE, 2011. 1051−1059. [doi: 10.1109/TrustCom.2011.144]

[4] Baruah SK, Chattopadhyay B, Li H, Shin I. Mixed-Criticality scheduling on multiprocessors. Real-Time Systems, 2014,50(1): 142−177. [doi: 10.1007/s11241-013-9184-2]

[5] Baruah SK, 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 (ESA). Springer-Verlag, 2011. 555−566. [doi: 10.1007/978-3-642-23719-5_47]

[6] Ekberg P, Yi W. Outstanding paper award: 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. [doi: 10.1109/ECRTS.2012.24]

[7] Baruah SK, Mok AK, Rosier LE. Preemptively scheduling hard-real-time sporadic tasks on one processor. In: Proc. of the 11th Real-Time Systems Symp. (RTSS). IEEE, 1990. 182−190. [doi: 10.1109/REAL.1990.128746]

[8] Baruah SK, Li H, Stougie L. Towards the design of certifiable mixed-criticality systems. In: Proc. of the Real-Time and Embedded Technology and Applications Symp. (RTAS). IEEE, 2010. 13−22. [doi: 10.1109/RTAS.2010.10]

[9] Baruah SK, Burns A, Davis RI. Response-Time analysis for mixed criticality systems. In: Proc. of the 32rd Real-Time Systems Symp. (RTSS). IEEE, 2011. 34−43. [doi: 10.1109/RTSS.2011.12]

[10] Baruah SK, Bonifaci V, D'Angelo G, Li H, Marchetti-Spaccamela A, Megow N, Stougie L. Scheduling real-time mixed-criticality jobs. IEEE Trans. on Computers, 2012,61(8):1140−1152. [doi: 10.1109/TC.2011.142]

[11] Mollison MS, Erickson JP, Anderson JH, Baruah SK, Scoredos JA. Mixed-Criticality real-time scheduling for multicore systems. In: Proc. of the 10th Int'l Conf. on Computer and Information Technology (CIT). IEEE, 2010. 1864−1871. [doi: 10.1109/CIT.2010.320]

[12] Pathan RM. Schedulability analysis of mixed-criticality systems on multiprocessors. In: Proc. of the 24th Euromicro Conf. on Real- Time Systems (ECRTS). IEEE, 2012. 309−320. [doi: 10.1109/ECRTS.2012.29]

[13] Li H, Baruah SK. Outstanding paper award: Global mixed-criticality scheduling on multiprocessors. In: Proc. of the 24th Euromicro Conf. on Real-Time Systems (ECRTS). IEEE, 2012. 166−175. [doi: 10.1109/ECRTS.2012.41]

[14] De Niz D, Lakshmanan K, Rajkumar R. On the scheduling of mixed-criticality real-time task sets. In: Proc. of the 30th Real-Time Systems Symp. (RTSS). IEEE, 2009. 291−300. [doi: 10.1109/RTSS.2009.46]

[15] Lakshmanan K, de Niz D, Rajkumar R, Moreno G. Resource allocation in distributed mixed-criticality cyber-physical systems. In: Proc. of the 30th Int'l Conf. on Distributed Computing Systems (ICDCS). IEEE, 2010. 169−178. [doi: 10.1109/ICDCS.2010.91]

[16] 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. [doi: 10.1109/RTAS.2011.13]

[17] Li H, Baruah SK. An algorithm for scheduling certifiable mixed-criticality sporadic task systems. In: Proc. of the 31th Real-Time Systems Symp. (RTSS). IEEE, 2010. 183−192. [doi: 10.1109/RTSS.2010.18]

[18] Guan N, Ekberg P, Stigge M, Yi W. Effective and efficient scheduling of certifiable mixed-criticality sporadic task systems. In: Proc. of the 32rd Real-Time Systems Symp. (RTSS). IEEE, 2011. 13−23. [doi: 10.1109/RTSS.2011.10]

[19] Liu CL, Layland JW. Scheduling algorithms for multiprogramming in a hard-real-time environment. Journal of ACM, 1973,20(1): 46−61. [doi: 10.1145/321738.321743]

[20] Audsley NC. Optimal priority assignment and feasibility of static priority tasks with arbitrary start times. Technical Report, YCS 164, Department of Computer Science, University of York, 1991.

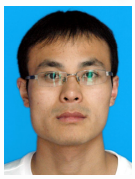
[21] Johnson DS. Near-Optimal bin packing algorithms [Ph.D. Thesis]. Boston: Massachusetts Institute of Technology, 1973.

谷传才(1982－),男,河北廊坊人,博士生,主要研究领域为实时系统,嵌入式系统.
E-mail: gucc.gu@gmail.com

王义(1961－),男,博士,教授,博士生导师,CCF 会员,主要研究领域为实时系统,嵌入式系统.
E-mail: yi@it.uu.se

关楠(1981－),男,博士,副教授,CCF 会员,主要研究领域为实时系统,嵌入式系统.
E-mail: guannan@ise.neu.edu.cn

邓庆绪(1970－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为嵌入式系统,物联网工程.
E-mail: dengqx@mail.neu.edu.cn



于金铭(1990－),男,硕士,主要研究领域为实时系统.
E-mail: yujinming521@126.com