引用格式: 李天森, 黄姝娟, 肖锋, 等. 混合关键系统半划分调度算法研究[J]. 微电子学与计算机, 2023, 40(3): 75-84. [LI T S, HUANG S J, XIAO F, et al. Research on semi-partition scheduling algorithm for mixed-criticality systems[J]. Microelectronics & Computer, 2023, 40(3): 75-84.] **DOI:** 10.19304/J.ISSN1000-7180.2022.0427

混合关键系统半划分调度算法研究

李天森[1], 黄姝娟[1], 肖 锋[1], 张文娟[2], 陈术山[1]
(1 西安工业大学 计算机科学与工程学院, 陕西 西安 710021;
2 西安工业大学 基础学院, 陕西 西安 710021)

摘 要: 混合关键系统是现代嵌入式系统发展的主要趋势之一,其中高关键任务代表紧急度高或者重要程度高的实际任务,往往需要优先保证. 为了保证高关键级别任务的执行,当前的混合关键任务调度算法中常常存在对低关键级别任务采用丢弃或者调度不及时的现象,造成在关键级别转换时,任务丢失时限率较大且系统利用率较低. 为此,本文在具有双重关键级别的混合关键系统中,对 EDF-os 半划分调度算法进行改进.首先,在划分阶段,将高关键级别的任务作为固定任务,低关键级别的任务按照利用率使用 Worst-Fit 策略进行划分. 其次,在执行阶段,采用 job 边界迁移形式,并详细讨论了在不同系统关键级别之下,不同关键级别任务优先级确定的策略,根据优先级对任务进行调度执行. 最后,模拟具有双关键级别的多处理器混合关键系统,随机产生任务集进行仿真实验,结果表明,该方法使得低关键级别任务的可执行比例平均提升了 14.8%,任务丢失时限率降低了 19.7%.

关键词: 任务调度; 混合关键系统; 半划分算法; 优先级; 多处理器
中图分类号: TP301.6 **文献标识码:** A **文章编号:** 1000-7180(2023)03-0075-10

Research on semi-partition scheduling algorithm for mixed-criticality systems

LI Tiansen[1], HUANG Shujuan[1], XIAO Feng[1], ZHANG Wenjuan[2], CHEN Shushan[1]
(1 School of Computer Science and Engineering, Xi'an Technological University, Xi'an 710021, China;
2 School of Science, Xi'an Technological University, Xi'an 710021, China)

Abstract: Mixed-criticality system is one of the main trends in the development of modern embedded systems. The high critical task represents the practical task with high urgency or importance, which usually needs to be guaranteed first. In order to ensure the execution of high-level critical tasks, the current mixed critical task scheduling algorithms often discard or schedule low-level critical tasks in a timely manner. As a result, the task loss time limit rate is large and the system utilization rate is low during the critical level conversion. Therefore, the EDF-os semi-partition scheduling algorithm is improved in hybrid critical systems with dual critical levels. First, in the division phase, the tasks at high critical levels are treated as fixed tasks and the tasks at low critical levels are divided by utilization using the Worst-Fit policy. Secondly, in the execution phase, the form of job boundary migration is used, and the strategies for determining the priority of tasks at different critical levels under different system critical levels are discussed in detail, and tasks are scheduled according to the priorities. Finally, a multi-processor hybrid critical system with dual critical levels is simulated, and task sets are randomly generated for simulation experiments. The results show that the proposed method increases the executable ratio of low-

收稿日期: 2022-07-17; **修回日期:** 2022-08-15
基金项目: 国家自然基金面上项目(62171361); 陕西省重点研发计划一般项目(2022GY-119); 陕西省科技厅自然科学基础研究计划(2021JM-440); 陕西省西安市未央区科技项目(201925)

critical level tasks by 14.8% on average, and decreases the task loss time rate by 19.7%.

Key words：task scheduling；mixed-criticality system；semi-partitioned algorithm；priority；multi-processor

1 引言

当前,嵌入式系统为了节省功耗,将越来越多日益复杂的功能需求集成在同一个硬件平台上. 这些功能需求具有不同的关键级别,从而形成不同关键级别下的实时周期任务. 而传统的实时周期任务调度方法因为没有考虑系统关键级别转化过程,已经不能适用该类混合关键系统,因此,需要考虑新的调度方法.

传统的实时周期任务调度方法可归纳为三类,分别是全局调度（Global Scheduling）、划分调度（Partitioned Scheduling）以及半划分调度（Semi-Partitioned Scheduling）[1]. 目前针对混合关键系统中任务调度算法的研究主要是对前两种调度方法的改进,虽然取得一定成果,但仍然会存在即使有空闲处理器核的情况下,低关键级别的任务要么被丢弃要么迁移开销较大,这直接导致了系统利用率不高,上下文开销大,时限丢失率增加[2]. 而传统的半划分调度算法是结合了全局调度和划分调度的优点而提出的,但由于其对划分策略有较高要求,目前在混合关键系统中应用较少. 本文在传统半划分调度算法 EDF-os（Earliest-Deadline-First-Based Optimal Semi-partitioned）[3] 的基础上进行改进,将其扩展到混合关键系统中,提出一种新的半划分调度算法—MC-EDF-os. 在该算法中将高关键级别任务和高利用率任务尽量划分为固定任务,低关键级别任务作为迁移任务按比例划分至不同处理器上. 当关键级别提升之后,在高关键级别任务所在处理器上的其他任务若无法继续执行,则根据 job 边界迁移规则将其迁移至就近有空闲资源的处理器核上,对于不受高关键级别任务影响的其他任务则保持原有划分状态,从而可以在一定程度上提高任务的可调度性.

2 相关工作

Vista 等人[4] 首次对混合关键系统模型进行详细介绍,并对任务在不同关键级别下的执行进行了分析,继而指出系统需要满足不同关键级别任务的执行需求. 随后 Baruah [5] 等人对经典的任务调度算法 EDF 进行分析,指出 EDF 在混合关键系统中并不适用,并提出一种新的算法——EDF-VD（Earliest Deadline First with Virtual Deadlines）为高关键级的任务设置更短的虚拟截止时限,当关键级切换之后,按照原始截止时限分配优先级,且丢弃低关键任务,从而保证高关键任务的需求. OCBP（Own-Criticality Based Priority）[6] 算法为典型的混合关键调度算法,随着处理器核数以及关键级别的增加,该算法的可调度性会随之下降,处理器性能无法被充分利用.

另一个典型的调度方法为 CAPA（Criticality as Priority Assignment）[7],该算法将任务关键级别作为优先级进行调度,导致低关键级别任务在多核环境下无法有效调度,系统利用率不高. 文献 [8] 指出,在实际性能对比之下,划分调度的可调度性以及系统开销往往要优于全局调度. 文献 [9] 中基于 EDF-VD 算法的基础上,提出一种关键感知的任务划分算法——CA-TPA(Criticality-Aware Task Partition Algorithm),通过分析不同关键级下任务的利用率对系统的影响,选择利用率增长最为缓慢的方式对任务进行划分,并通过理论推导出处理器负载失衡因子以及阈值,从而更大程度的保证任务的可调度性. 文献 [10] 中提出一种概率混合关键任务模型,为高关键级别任务设置任务过载概率参数,并结合 EDF-VD 算法优化最早截止期,通过概率计算不同阶段参数的期望值,最后提出一种基于概率的任务划分算法并通过实验验证. 划分调度虽然表现较为优秀,但在多核处理器上要么开销太大要么系统利用率较低. 为此,半划分调度方法在划分调度的基础上允许少量任务迁移到资源较为充足的处理器核上,从而减少全局调度的开销并提高了系统利用率. Zeng[11] 等人在 EDF-VD 算法的基础上使用 Worst-Fit 和 Best-Fit 等划分策略优先划分高关键任务,之后再对低关键任务进行划分,在执行阶段按照 EDF-VD 算法确定任务的优先级,若关键级切换之后,则丢弃低关键级别任务,从而保证高关键级别任务的执行需求. 文献 [12] 中对半划分算法在混合关键系统中迁移形式进行研究,提出了在关键级别切换时任务迁移的不同路线,并且通过概率的方式计算出应该进入高关键级别的处理器数量,并通过分析讨论得出一个较为通用的混合关键系统半划分调度模型. 文献 [13] 在 EKG[14] 算法的基础上提出一种新的半划分算法,在划分阶段按照 First-Fit 和 Worst-Fit 策略对任务进行划分. 在执行阶段把每个执行区间分为三部分,两端的区间用于执行迁移任务的第一、

二部分,中间的时段用于执行固定任务. 在实验过程中发现,该算法可能会出现关键级翻转问题,又通过设置合适的虚拟截止期对该算法进行改进. 然而这些算法中迁移任务被高优先任务抢占后,则会立即迁移至其他具有空闲资源的处理器上,相较于任务边界限制的迁移形式会产生更多的迁移开销.

3　混合关键系统模型

本文主要研究双关键级别的混合关键任务模型,即在具有M个同构处理器$P=\{P_1, P_2, \cdots, P_m\}$(各个处理器间共享存储器,即采用对称多处理器体系结构)的混合关键系统中包含n个周期性任务集合$T=\{\tau_1, \tau_2, \cdots, \tau_n\}$,每个任务可由一个四元组$\tau_i=(\chi_i, C_i(\mathrm{LO}), C_i(\mathrm{HI}), T_i)$表示,其中:

χ_i表示任务的关键级别,$\chi_i=\mathrm{LO}$时表示该任务为低关键级别任务,$\chi_i=\mathrm{HI}$时则表示该任务为高关键级别任务.

$C_i(\mathrm{LO})$和$C_i(\mathrm{HI})$分别表示任务在对应关键级别下最坏执行时间,当$\chi_i=\mathrm{LO}$时,$C_i(\mathrm{LO})=C_i(\mathrm{HI})$,当$\chi_i=\mathrm{HI}$时,$C_i(\mathrm{LO})<C_i(\mathrm{HI})$.

T_i表示任务的周期,为正整数,表示任务τ_i中连续两个job间最大时间间隔.

任务在执行前需要将其划分至各个处理器上,记P_i^m表示任务τ_i划分至处理器P_m上,任务在开始执行时系统处于低关键模式,此时所有任务都按照$C_i(\mathrm{LO})$执行,若有高关键任务的实际执行时间超过对应的$C_i(\mathrm{LO})$,则关键级别切换至高关键模式,之后高关键级别任务按照$C_i(\mathrm{HI})$执行,而低关键级别任务的执行时间不变. 将任务对应的利用率用u_i表示,其中$u_i(\mathrm{LO})=C_i(\mathrm{LO})/T_i$,$u_i(\mathrm{HI})=C_i(\mathrm{HI})/T_i$,当系统关键级处于低关键模式时,所有任务的利用率之和表示为:

$$U_{\mathrm{LO}}=\sum_{\tau_i\in T}u_i(\mathrm{LO}) \tag{1}$$

当系统关键级切换至高关键模式,任务的利用率之和发生变化,由于高关键级别任务的执行时间发生变化,对应的任务利用率也发生变化,而低关键级别任务的利用率则不发生变化,此时任务的利用率之和可表示为:

$$U_{\mathrm{HI}}=\sum_{\tau_i\in T\cap\chi_i=\mathrm{LO}}u_i(\mathrm{LO})+\sum_{\tau_i\in T\cap\chi_i=\mathrm{HI}}u_i(\mathrm{HI}) \tag{2}$$

4　算法描述

4.1　相关定义及算法介绍

本文在已有的半划分算法 EDF-os 基础上进行改进,相关定义如下:

定义 1:　任务τ_i在处理器P_m上所占的份额$s_{i,m}$[15]表示为:

$$\sum_{1\leqslant m\leqslant M}s_{i,m}=u_i \tag{3}$$

式中,u_i表示任务τ_i的利用率,若任务τ_i为固定任务,则$s_{i,m}=u_i$;若任务τ_i为迁移任务,且分配到的处理器为$P_1, \cdots, P_m$,则$u_i=s_{i,1}+\cdots+s_{i,m}$,由此可得$s_{i,m}\leqslant u_i$是恒成立的.

定义 2:　处理器P_m能够处理任务τ_i的工作负载$f_{i,m}$[15]表示为:

$$f_{i,m}=s_{i,m}/u_i \tag{4}$$

定义 3:　固定任务: 若任务τ_i对应的$s_{i,m}=u_i$,即$f_{i,m}=1$时,表示该任务能够完全划分至处理器P_m上,就称该任务为固定任务.

定义 4:　迁移任务:　若任务τ_i对应的$s_{i,m}<u_i$,且$f_{i,m}$不为 1 时,表示该任务无法完全划分至处理器P_m上,此时该任务就为迁移任务,还需另一个处理器提供一部分资源.

EDF-fm[15]算法和 EDF-os[3]算法是两种比较典型的基于 job 边界限制的半划分调度算法,相较于非 job 边界迁移,前者因无需在不同任务的 job 之间进行上下文切换而节省大量开销,从而在软实时系统中具有更优的性能. 半划分算法主要由两部分组成,一是任务划分阶段,在此阶段,通常是离线处理,即静态的按照一定的规则先对任务进行划分. 二是在线任务执行阶段,划分好的任务按照优先级调度执行. 由于上述两种算法都是针对一般软实时周期任务,且 EDF-os 较优,但它们都不适合混合关键系统,因此,本文将对 EDF-os 算法进行改进.

首先对 EDF-os 算法思想进行简要介绍.

(1)任务划分阶段

在划分之前,首先将所有任务按照利用率降序排列,然后选择与处理器核数量相等的前 M 个任务依次划分至 M 个处理器上. 根据定义 3 可知,这些任务都将划分为固定任务. 之后,将剩余任务依次划分至有空闲资源的处理器上,若有任务无法完全划分至某个处理器上,根据定义 1 和定义 4 计算在不同处理器上所需的份额,选择合适的处理器进行划分.

(2)任务执行阶段

在执行阶段,EDF-os 算法中规定,同一处理器上的迁移任务的优先级要高于固定任务. 若该处理器上有两个或两个以上的迁移任务,选择该处理器为非第一个分配的处理器的任务以最高优先级执行. 换句话说,任务τ_i为迁移任务,且划分到的处理器为P_m和

P_{m+1},而处理器P_{m+1}上还有另外一个迁移任务τ_{i+1},它所分配到的处理器为P_{m+1}和P_{m+2},则在处理器P_{m+1}上任务τ_i比τ_{i+1}具有更高的优先级.

EDF-os 算法在传统实时任务调度中被验证具有最优性能,若不加改进直接运用到混合关键系统中,则会存在一些弊端. 以一个示例进行说明.

表 1 中包含 6 个任务,其中τ_1,τ_2为高关键级别任务,其余为低关键级别任务. 假设现按照 EDF-os 算法的思想对混合关键任务进行划分,图 1 和图 2 分别表示低关键模式和高关键模式下任务的划分结果. 图 1 和图 2 中颜色相同的为同一任务,从划分结果来看,高关键模式和低关键模式下任务的划分差异较大,显然当关键级别发生变换时,由于高关键级别任务的利用率增大,原来的排列顺序发生变化,若依旧按照任务利用率进行划分,则会产生较多不必要的开销. 因此若 EDF-os 直接运用至混合关键系统中,在任务划分策略上显然有着明显的不足.

表 1 任务示例

Tab. 1 Task instances

任务号	释放时间	周期	关键级	c_i(LO)	c_i(HI)
τ_1	0	5	2	3	4
τ_2	0	10	2	6	7
τ_3	0	8	1	5	5
τ_4	0	6	1	4	4
τ_5	0	10	1	5	5
τ_6	0	12	1	8	8

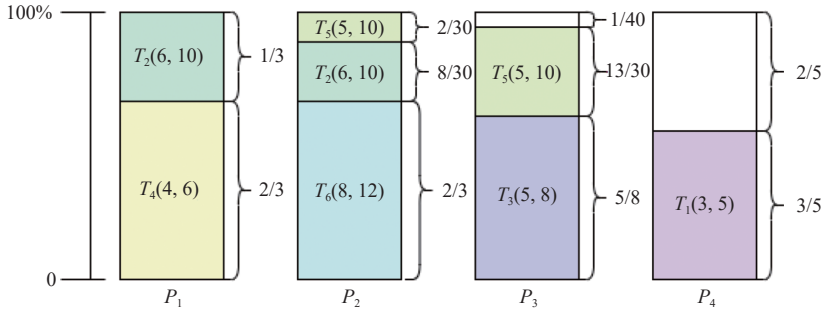

图 1 EDF-os 低关键模式下任务划分结果

Fig. 1 Task assignment results of EDF-os algorithm in low criticality

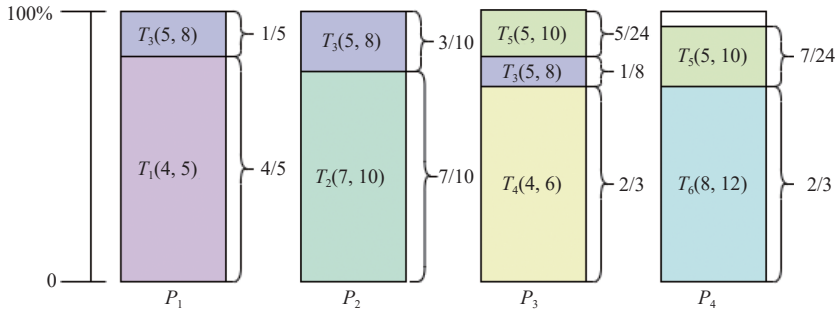

图 2 EDF-os 高关键模式下任务划分结果

Fig. 2 Task assignment results of EDF-os algorithm in high criticality

因此本文在 EDF-os 算法的基础上,优先划分高关键级别任务到单独的处理器上(假设高关键级别任务数量小于等于处理器数量),再将剩余任务按照一定规则依次划分. 当关键级别变化时,只需将高关键级别任务实际执行时间切换为c_i(HI),在高关键级别任务所在处理器上的迁移任务可能会面临资源不足无法执行的情况,此时,需要根据迁移规则将这些任务迁移至空闲处理器上执行.

4.2 改进的混合关键半划分算法

改进后的半划分调度算法 MC-EDF-os 也分为两个阶段.

(1)任务划分阶段

在该阶段不同于 EDF-os 算法直接按照任务利用率降序排列,首先将高关键级别任务各自划分至一个空闲处理器上,之后将剩余任务按照利用率降序排列并计算对应的$s_{i,m}$和$f_{i,m}$,优先选择还未划分任务的

处理器,依次划分至各个处理器上,直到任务划分完毕. 具体划分阶段的流程图如图 3 所示. 同样以表 1 中示例为例,对表中任务进行划分,其结果如图 4、图 5 所示.

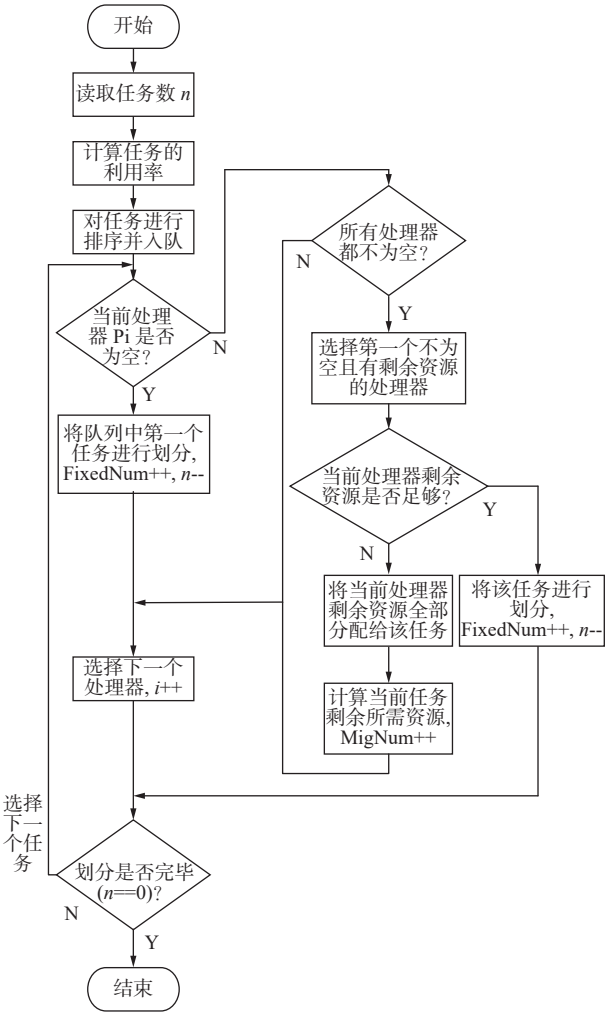

图 3　任务划分流程

Fig. 3 Task assignment flowchart

从划分结果可以看出,当关键级发生变换时,在低关键模式下划分好的固定任务基本没有变化,只是高关键级别任务由于 WCET 增加,从而占有了更多的 CPU 资源,导致原本划分在该处理器上的任务因剩余资源不够不得不迁移至其他有剩余资源的处理器上.

在实际任务划分过程中,可能会存在这样一种情况：即低关键模式下任务都能完全划分,而在关键级升高后,由于高关键任务利用率增加,使得任务集的总利用率也随之增加,导致任务不能完全划分至处理器上. 若出现这种情况,为了避免任务划分直接失败,丢弃利用率最低的一个或几个低关键任务,从而保证高关键任务能够正确划分并执行. 而这种有选择的丢弃部分低关键任务保证系统的正常执行在混合关键系统中是可接受的[11].

（2）任务执行阶段

在执行阶段,主要是确定每个处理器上任务的优先级顺序,选择合适的执行顺序也是算法可调度性好与坏最直接的体现. 混合关键系统中,高关键任务较为重要,如果执行出错可能会产生比较严重的后果. 因此任务的优先级按以下规则进行排序：

● 低关键模式下：

（1）若在同一处理器上同时有固定任务和迁移任务,则迁移任务的优先级高于固定任务；

（2）若同一处理器上有多个迁移任务或者多个固定任务. 其中多个迁移任务选择该处理器为非第一个分配的处理器的任务以最高优先级执行；多个固定任务之间则按照 EDF 算法（截止时限越早优先级越高）进行优先级确定.

● 高关键模式下：

（1）同一处理器上若固定任务中存在高关键级别任务,则其优先级升为最高,对于同为低关键级别任务或者同为高关键级别任务,按照最早截止时限确定优先级；

（2）同一处理器上的多个迁移任务之间,仍然

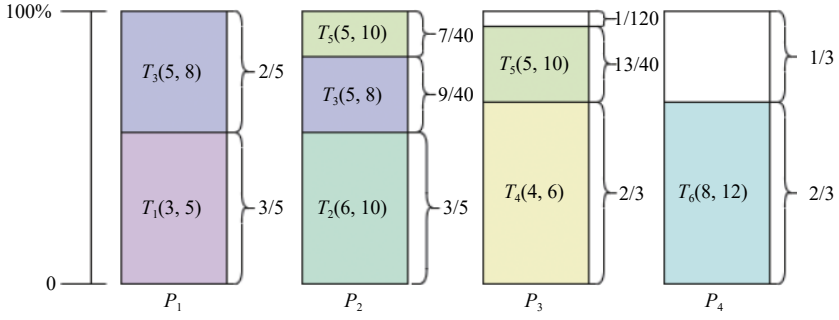

图 4　MC-EDF-os 算法低关键模式下任务划分结果

Fig. 4 Task assignment results of MC-EDF-os algorithm in low criticality

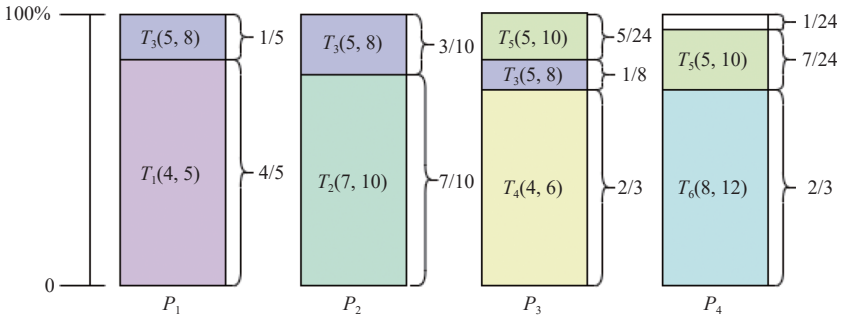

图 5　MC-EDF-os 算法高关键模式下任务划分结果

Fig. 5 Task assignment results of MC-EDF-os algorithm in high criricality

选择该处理器为非第一个分配的处理器的任务以最高优先级执行；

（3）若同一处理器上既有固定任务（低关键任务）又有迁移任务,则迁移任务优先级高于固定任务.

具体执行阶段流程如图 6.

5　实验结果及分析

实验主要分为三个部分,分别为任务集生成、控制变量进行对比实验以及实验结果分析.

5.1　任务集生成

混合关键任务集的生成采用文献 [16] 中的方法,我们对其进行修改并生成如下几个主要参数：

（1）任务集利用率U_i：使用 UUnifast 算法[17]随机生成均匀分布的任务集利用率,其值范围为 [0.4,4.0],步长为 0.4,设置低关键级别任务的利用率范围为 [0.05,0.25]；

（2）周期T_i：任务周期为正整数,其值取值范围为 [10,100](ms) 随机生成；

（3）低关键级别下任务的最坏执行时间C_i(LO)：C_i(LO)的值根据周期和任务的利用率计算得出,$C_i(\mathrm{LO}) = u_i(\mathrm{LO}) * T_i$；

（4）高关键级别下任务的最坏执行时间C_i(HI)：$C_i(\mathrm{HI}) = CF * C_i(\mathrm{LO})$（$CF = 1.5$）；根据给定的利用率范围,每个利用率随机产生 10 个任务集,每个任务集中包括 6～20 个利用率不同的任务并生成 job 实例. 最后取这 10 个任务集运行结果的平均值作为最后的实验结果.

5.2　实验对比及结果分析

为验证本文算法的有效性,选取 CAPA[7] 算法、EDF-VD[5] 算法以及 EDF-os[3] 算法作为对比算法,在 4 核处理器平台上进行仿真实验,并从任务的可调度性,高低关键任务可调度率以及任务丢失时限率等指标进行对比.

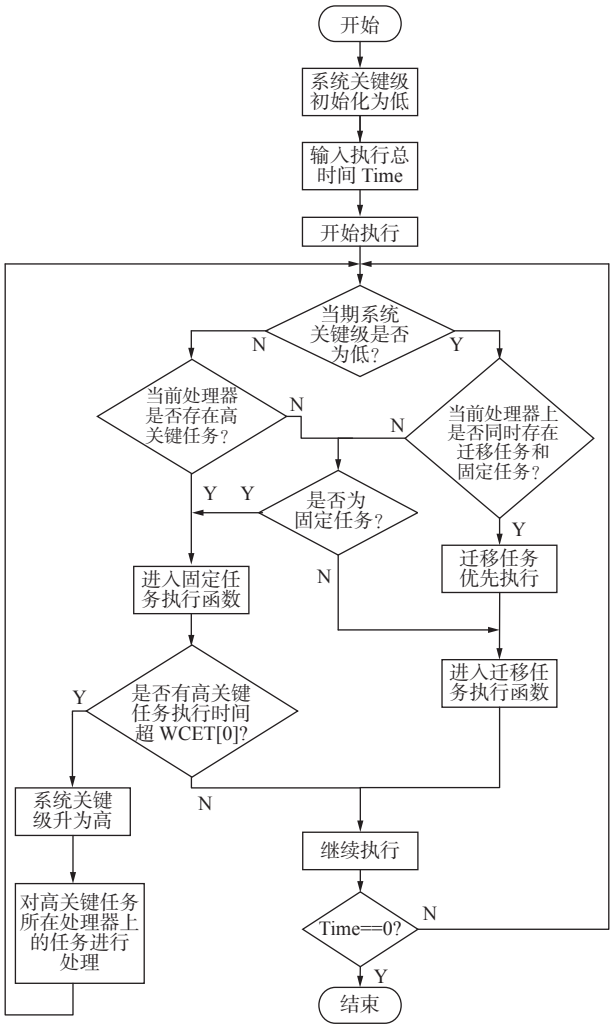

图 6　任务执行流程图

Fig. 6 Task execution flowchart

（1）任务的可调度性验证

任务的可调度性表示在任务执行过程中,正常完成任务的总数占生成任务总数的比率,又称为任务可接受率（Acceptance Ratio, AR）.

图 7 表示不同利用率对应的任务集在调度过程中实际执行的情况,x 轴表示任务集规范化（任务集

总利用率/处理器核数）后的利用率. 从图 7 中可以看出,任务集利用率在 0.4（规范前为 1.6）之前,四种调度算法中任务的可调度性都较高,而从 0.4 开始,不同算法的性能开始有所差异. 随着任务集利用率的增长,CAPA 算法和 EDF-VD 算法中任务的可调度率下降速度较为迅速. 这是因为 CAPA 算法和 EDF-VD 算法在任务执行过程中,一旦发生系统关键级别切换,两者优先保证高关键任务的执行需求,导致低关键任务无法得到有效的执行. 本文算法中,在任务执行之前,就将任务划分到指定处理器上,在执行过程中,任务按照预先划分的结果在固定处理器上执行,且在关键级切换后,只对受影响的低关键任务进行再划分,对于每个处理器上执行的任务数有限且固定,大大减少了不同任务相互影响而产生的开销,进而提升每个处理器的利用率,任务总的可调度性也随之提升. 而 EDF-os 算法虽然将任务划分到指定处理器上,但是在关键级切换时,由于高关键任务的利用率增加,可能会导致之前划分的任务需要完全重新划分,在这过程中浪费了一定资源. 因此对于任务的调度性会有影响.

（2）任务抢占次数与迁移次数的对比

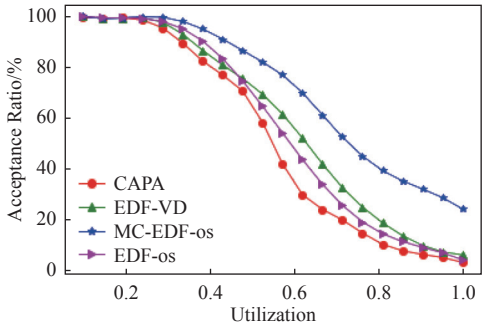

图 7　任务可调度性随任务集利用率变化的关系

Fig. 7 The relationship between task schedulability and task set utilization

任务抢占次数与迁移次数是任务调度中比较重要的评价指标,过多的抢占和迁移会导致算法的有效性降低.

表 2 表示四种算法在不同任务集利用率下成功调度的 200 次作业中对应的抢占次数和迁移次数的对比结果. 随着利用率的增加,抢占次数和迁移次数也是逐渐增加,在任务集利用率为 0.7（规范化前对应 2.8）时,两者达到最大值. 当任务集利用率增加时,任务抢占和迁移会随着任务实际执行时间的增加而增加.

表 2　任务抢占次数与迁移次数

Tab. 2 Preemptions and migrations of tasks

利用率	抢占次数				迁移次数			
	CAPA	EDF-VD	MC-EDF-os	EDF-os	CAPA	EDF-VD	MC-EDF-os	EDF-os
0.1	2	1	1	1	2	1	0	0
0.2	3	2	1	1	3	2	1	1
0.3	8	6	4	5	7	4	3	4
0.4	24	14	8	9	18	10	7	9
0.5	36	28	16	22	32	27	16	21
0.6	43	30	16	36	38	33	16	34
0.7	54	40	22	49	45	40	21	43
0.8	41	32	22	37	32	28	21	30
0.9	33	25	17	30	28	22	16	25
1.0	28	20	16	25	26	19	15	22

而当任务集利用率达到 0.8（规范化前为 3.2）之后,整个处理器的资源会变得明显紧张,会有大量任务因为错过截止时间而丢失,反而任务的抢占和迁移会下降. 但整体来说,CAPA 算法和 EDF-VD 算法在任务执行过程中,任务发生抢占之后有很大概率迁移到别的处理器上执行,会产生更多的迁移开销. 本文算法中,只有迁移任务会发生这种情况,因此可以有效避免更多的迁移开销. 而 EDF-os 算法在任务集利用率较小时与本文算法相差无几,但在任务集利用率不断增加（0.6 以后）,其抢占和迁移次数增多,表

现甚至不如 EDF-VD 算法,由于关键级切换后,可能需要对全部任务进行重新划分,迁移任务相比于关键级切换之前变化较大,因此会增加更多的迁移开销.

（3）高低关键级别任务可调度率对比

混合关键任务调度中,最重要的是保证高关键任务的执行需求,因此现有的很多算法对于低关键级别任务采取消极应对策略. 但低关键级别任务同样会对整个系统产生一定影响,因此,需要对低关键级别任务的可调度性进行研究. 图 8 和图 9 分别表示四种算法中低关键级别任务和高关键级别任务的可调度性.

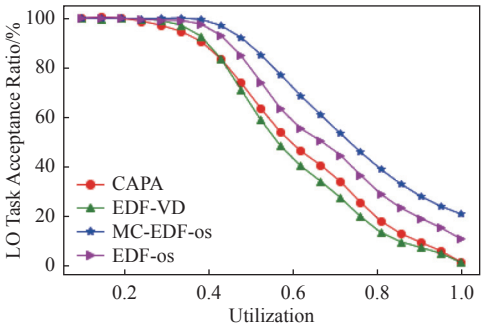

图 8　低关键任务可调度率

Fig. 8 schedulable of low criticality tasks

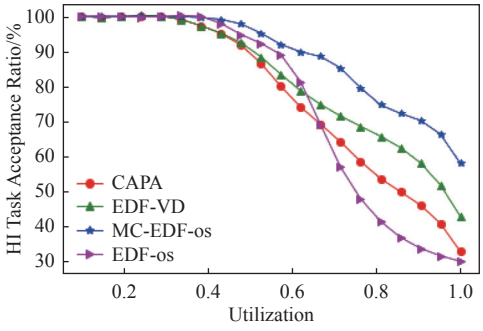

图 9　高关键任务可调度率

Fig. 9 The schedulable rate of high criticality tasks

从图 9 中可以看出,CAPA、EDF-VD 和 MC-EDF-os 三种算法对于高关键级别任务的执行需求相对较好,而 EDF-os 算法在时间执行过程中并未对高关键任务进行优先处理,因此随着任务集利用率的不断增加,当关键级切换之后,高关键任务可能由于利用率更高,而划分为固定任务,优先级会低于迁移任务（大部分为低关键任务）,因此,高关键任务的可调度性较差. 由此可见,改进算法对于高关键任务的支持较好,而且更加符合混合关键系统的特点. 而图 8 中低关键级别任务的可调度性则有明显的差异. 当任务集利用率在 0.4 之后,CAPA 算法和 EDF-VD 算法中低关键级别任务丢失的数量迅速增加,这也正好印证了两种算法对应的优缺点,都优先保证高关键级别任务的执行需求,且 EDF-VD 算法在系统关键级切换时,将低关键级别任务挂起,只执行高关键级别任务,因此该算法中低关键级别任务的可调度性更低,但是对高关键级别任务却有更高的调度性. 相较而言,EDF-os 算法和本文算法中,虽然随着利用率的增加,低关键级别的任务可调度率也随之下降,但总体要优于 CAPA 和 EDF-VD 算法,这是因为在离线阶段就以静态的方式对任务进行划分,在执行阶段对于每个任务的执行过程也能追踪溯源,充分地利用了处理器的资源,从而能够调度更多的任务.

（4）任务丢失时限率验证

任务丢失时限率表示所调度的任务中,执行过程中丢失时限（执行时错过截止时限）的任务占所调度任务总数的比率.

图 10 表示随着任务数的增加,四种算法中任务丢失时限率的比较. 可以看出任务总数在 60 之前,由于 CPU 资源较为充足,四种算法对应的任务丢失时限的比率都较低. 在这之后,随着任务数不断增加,不同算法对应的任务丢失时限率差异逐渐变大. 其中 CAPA 算法和 EDF-VD 算法中,由于只需保证高关键级别任务的执行需求. 当任务总数增加时,高关键级别任务的利用率随之增加,CPU 资源不足以调度全部任务,且全局调度算法中,任务的执行过程难以预测,因此两者都是挂起或者丢弃低关键级别任务从而给高关键级别任务留以充足的资源,进而任务丢失的数量不断增加. 相对于 CAPA 算法和 EDF-VD 算法,EDF-os 算法虽然在关键级切换后,由于划分过程中没有对高关键任务进行充分考虑,会有更多的高关键任务因处理器资源不足无法调度执行,但是在低关键模式下,其任务的可调度性较好. 而本文算法在执行前已将不同关键级任务划分到指定处理器上,且迁移任务

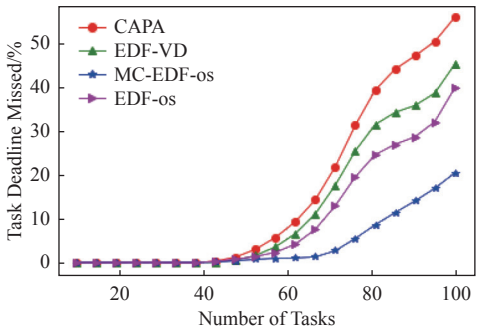

图 10　任务丢失时限率

Fig. 10 Task deadline missed rate

可以在不同处理器上进行调度,关键级切换后,不需要对所有任务进行重新划分,充分的利用了处理器的资源,大大降低了任务丢失时限的数量.

6　结束语

本文在半划分调度算法 EDF-os 进行改进,通过结合混合关键系统的特点,提出了一种适用于混合关键系统的半划分调度算法 MC-EDF-os,为保证高关键任务的执行需求,优先将高关键任务划分为固定任务且各自分配一个单独的处理器,若系统关键级发生变化,则更新高关键级别任务的实际执行时间,而对于该处理器上的其他任务,若无法完全划分在该处理器上,则就近迁移. 而对于其他不受高关键级别任务影响的低关键级别任务则按照原来的划分继续执行. 通过实验验证,本文算法在任务的可调度性及低关键级别任务的执行比例方面具有较好的效果.

在实际中,由于任务具有更复杂的功能,往往系统中有多个关键级,任务的数量也趋向于更为庞大,且高关键级别任务数量也随之增多. 本文中设定高关键级别任务的数量小于等于处理器的个数,这在实际应用中有一定的限制. 因此下一步期望在具有多个关键级的混合关键系统中继续研究具有多个高关键级别任务的调度策略. 另外,本文中对所提算法的可调度性分析未做详细讨论,由于篇幅原因,拟在后续工作中继续讨论.

参考文献:

[1] 赵雪静, 蔡加豪. 混合关键性多核系统调度综述[J]. 电脑与信息技术, 2016, 24(2): 16-20. DOI: 10.19414/j.cnki.1005-1228.2016.02.005.
ZHAO X J, CAI J H. Mixed criticality multicore systems scheduling[J]. Computer and Information Technology, 2016, 24(2): 16-20. DOI: 10.19414/j.cnki.1005-1228.2016.02.005.

[2] 谷传才, 关楠, 于金铭, 等. 多处理器混合关键性系统中的划分调度策略[J]. 软件学报, 2014, 25(2): 284-297. DOI: 10.13328/j.cnki.jos.004534.
GU C C, GUAN N, YU J M, et al. Partitioned scheduling policies on multi-processor mixed-criticality systems[J]. Journal of Software, 2014, 25(2): 284-297. DOI: 10.13328/j.cnki.jos.004534.

[3] ANDERSON J H, ERICKSON J P, DEVI U M C, et al. Optimal semi-partitioned scheduling in soft real-time systems[J]. Journal of Signal Processing Systems, 2016, 84(1): 3-23. DOI: 10.1007/s11265-015-0983-7.

[4] VESTAL S. Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance[C]//28th IEEE International Real-Time Systems Symposium. Tucson: IEEE, 2007: 239-243. DOI: 10.1109/RTSS.2007.47.

[5] BARUAH S, BONIFACI V, DANGELO G, et al. The preemptive uniprocessor scheduling of mixed-criticality implicit-deadline sporadic task systems[C]//2012 24th Euromicro Conference on Real-Time Systems. Pisa: IEEE, 2012: 145-154. DOI: 10.1109/ECRTS.2012.42.

[6] BARUAH S, LI H H, STOUGIE L. Towards the design of certifiable mixed-criticality systems[C]//16th IEEE Real-time and Embedded Technology and Applications Symposium. Stockholm: IEEE, 2010: 13-22. DOI: 10.1109/RTAS.2010.10.

[7] DE NIZ D, LAKSHMANAN K, RAJKUMAR R. On the scheduling of mixed-criticality real-time task sets[C]//30th IEEE Real-time Systems Symposium. Washington: IEEE, 2009: 291-300. DOI: 10.1109/RTSS.2009.46.

[8] BARUAH S, CHATTOPADHYAY B, LI H H, et al. Mixed-criticality scheduling on multiprocessors[J]. Real-Time Systems, 2014, 50(1): 142-177. DOI: 10.1007/S11241-013-9184-2.

[9] HAN J J, TAO X, ZHU D K, et al. Criticality-aware partitioning for multicore mixed-criticality systems[C]//45th International Conference on Parallel Processing. Philadelphia: IEEE, 2016: 227-235. DOI: 10.1109/ICPP.2016.33.

[10] ZENG L N, XU C, LI R F. Partition and scheduling of the mixed-criticality tasks based on probability[J]. IEEE Access, 2019, 7: 87837-87848. DOI: 10.1109/ACCESS.2019.2926299.

[11] ZENG L N, LEI Y, LI Y X. A semi-partition algorithm for mixed-criticality tasks in multiprocessor platform[C]//2019 IEEE 10th International Conference on Software Engineering and Service Science (ICSESS). Beijing: IEEE, 2019: 694-697. DOI: 10.1109/ICSESS47205.2019.9040792.

[12] XU H, BURNS A. A semi-partitioned model for mixed criticality systems[J]. Journal of Systems and Software, 2019, 150: 51-63. DOI: 10.1016/j.jss.2019.01.015.

[13] YANG C H, WANG H, ZHANG J, et al. Semi-partitioned scheduling of mixed-criticality system on multi-processor platforms[J]. The Journal of Supercomputing, 2022, 78(5): 6386-6410. DOI: 10.1007/s11227-021-04101-y.

[14] ANDERSSON B, TOVAR E. Multiprocessor scheduling with few preemptions[C]//12th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications. Sydney: IEEE, 2006: 322-334. DOI: 10.1109/RTCSA.2006.45.

[15] ANDERSON J H, BUD V, DEVI U M C. An EDF-

based restricted-migration scheduling algorithm for multi-processor soft real-time systems[J]. Real-Time Systems, 2008, 38(2): 85-131. DOI: 10.1007/s11241-007-9035-0.

[16] BARUAH S K, BURNS A, DAVIS R I. Response-time analysis for mixed criticality systems[C]//2011 IEEE 32nd Real-time Systems Symposium. Vienna: IEEE, 2012: 34-43. DOI: 10.1109/RTSS.2011.12.

[17] BINI E, BUTTAZZO G C. Measuring the performance of schedulability tests[J]. Real-Time Systems, 2005, 30(1-2): 129-154. DOI: 10.1007/s11241-005-0507-9.

作者简介:

李天森 男,(1996-),硕士研究生. 研究方向为嵌入式系统、混合关键任务调度.

黄姝娟(通讯作者) 女,(1975-),博士,副教授.研究方向为嵌入式系统. E-mail: huangshujuan@xatu.edu.cn.

肖 锋 男,(1976-),博士,教授. 研究方向为智能信息处理.

张文娟 女,(1980-),博士,副教授.研究方向为机器学习.

陈术山 男,(1998-),硕士研究生. 研究方向为嵌入式系统、异构多核.