软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2015,26(Suppl.(2)):257−262

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

一种计算混合关键任务响应时间的方法*

白恩慈, 张伟哲

(哈尔滨工业大学 计算机科学与技术学院,黑龙江 哈尔滨 150001)

通讯作者: 张伟哲, E-mail: wzzhang@hit.edu.cn

摘 要: 混合关键系统中不同关键等级的任务在同一个平台运行,任务的可调度性分析更加复杂.基于目前最有效的固定优先级混合关键的调度算法 AMC(adaptive mixed criticality),提出了一种任务响应时间分析算法 AMC-PM(AMC partition max).该算法将任务最长执行时间(worst case execution time,简称 WCET)分成低关键等级态执行时间与高关键等级态执行时间,将这两部分对应的最长响应时间加起来得到总的响应时间上界.通过仿真实验,与已有的 AMC 响应式分析算法进行比较,结果表明,在任务高关键下最长执行时间较小时,与 AMC-rtb 相比,AMC-PM 能够显著地提高系统的可调度性.同时与 AMC-max 相比,AMC-PM 能够显著降低算法的运行时间.

关键词: 混合关键系统;自适应混合关键调度;可调度性;响应时间分析

中文引用格式: 白恩慈,张伟哲.一种计算混合关键任务响应时间的方法.软件学报,2015,26(Suppl.(2)):257−262. http://www.jos.org.cn/1000-9825/15036.htm

英文引用格式: Bai EC, Zhang WZ. Method of response-time analysis for mixed criticality systems. Ruan Jian Xue Bao/Journal of Software, 2015,26(Suppl.(2)):257−262 (in Chinese). http://www.jos.org.cn/1000-9825/15036.htm

Method of Response-Time Analysis for Mixed Criticality Systems

BAI En-Ci, ZHANG Wei-Zhe

(School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China)

Abstract: In mixed criticality systems, tasks with different criticality levels share a common platform, which makes the schedulability more complex. Considering AMC (adaptive mixed criticality) scheduling is currently the most effective fixed priority approach for scheduling mixed criticality systems, this work presents a response time analysis algorithm AMC-PM (AMC partition max) for AMC. In AMC-PM, the WCET (worst case execution time) of the task is partitioned into low critical execution time and high critical execution time. Then an upper bound of response time can be derived by adding the response times of the two parts together. For tasks with small WCET, evaluations illustrate that AMC-PM can significantly enhance the schedulability comparing with AMC-rtb and that AMC-PM can effectively decrease the run time comparing with AMC-max.

Key words: mixed-criticality system; adaptive mixed criticality scheduling; schedulability; response time analysis

随着嵌入式实时系统在日常生活中的广泛应用,不同安全关键等级的任务都需要用实时系统来处理.为了保证高关键等级任务不受低关键等级任务的影响,它们一般需要用不同的微处理器分别处理.这样一来,在成本、功耗、体积、质量等方面都给系统设计带来非常大的负担,同时也增加了系统的复杂性.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.另一方面,处理器性能的不断提升,使得不同关键等级的任务用同一硬件平台处理成为可能,这也成为解决上述问题的途径.

混合关键系统是允许不同关键等级的应用程序在相同的硬件平台上并存和交互的实时系统.自 2007 年 Vestal[1]首先形式化定义了混合关键性系统(mixed-criticality system)中的实时调度问题以来,混合关键性系统的

* 基金项目: 国家自然科学基金(61173145, 61472108); 教育部高等学校博士学科点专项科研基金(20132302110037)
收稿时间: 2015-08-07; 定稿时间: 2015-10-12

研究迅速成为近年来实时系统领域的热点问题.2010 年,Baruah 等人[2]提出了用于调度有限混合关键性任务的 OCBP(own criticality based priority)算法.同年,Li 等人[3]将 OCBP 算法扩展到偶发实时任务模型.OCBP 算法根据每个任务本身的关键性为每个任务分配一个优先级,根据优先级进行调度,缺点是可能导致系统的利用率较低.2011 年,Baruah 等人[4]又提出了混合关键任务的 AMC(adaptive mixed criticality)调度算法.AMC 算法系统刚开始设为低关键等级状态,一旦有任务的运行时间超过分配的时间上限,系统就自动转为高关键等级状态,此时,低关键等级任务不再运行.由于 AMC 算法在固定优先级混合关键任务调度中有较高的利用率和灵活性,2013 年,Gu 等人[5]将 Preemption Threshold 算法和 AMC 算法结合起来,给出了 PT-AMC 调度算法.该算法提高了任务的可调度性,减少了任务间的抢占.2014 年,Burns 等人[6]将 Deferred Preemption 算法与 AMC 算法结合起来,也提高了任务的可调度性,并减少了任务间的抢占.

混合关键任务的可调性分析是实时任务的调度的关键,其中基于响应时间分析(response-time analysis)的可调度性分析是非常重要的方法.正如文献[4]指出的那样,求解任务确切的响应时间的困难之处在于不存在一个临界时刻(critical instant)使得任务的响应时间最长,需要考虑各任务开始时间的不同组合.即使计算周期性任务的响应时间也是 NP 难的[7],因此 Baruah 等人在文献[4]中只给出两种求解任务响应时间上界的方法:AMC-rtb 算法和 AMC-max 算法.AMC-rtb 算法复杂性较低,但是求解的响应时间的值较大,因此可调度性较低.相反,AMC-max 算法复杂性较高,但是可调性较高.本文提出一种新的求解任务响应时间上界的方法——AMC-PM (AMC partition max)算法,该算法可调性比 AMC-rtb 算法高,同时复杂性比 AMC-max 算法低.

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 AMC-PM 算法.第 3 节通过仿真实验对比本文提出的 AMC-PM 算法与 AMC-rtb 算法和 AMC-max 算法的性能.第 4 节对全文做简要的 总结.

1 系统模型与定义

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.我们研究的是一组包含 N 个独立偶发任务的任务集 $\Gamma=\{\tau_1,\tau_2,...,\tau_N\}$.每个任务 τ_i 可以由一个四元组来定义,即 $\tau_i=\{T_i,D_i,\vec{C}_i,L_i\}$.其中,$T_i$ 表示任务 τ_i 任意两个连续作业间的最小时间间隔.D_i 表示任务 τ_i 的相对截止时间.$\vec{C}_i$ 是一个向量,对应着任务 τ_i 不同关键等级下的最长执行时间(worst case execution time,简称 WCET),且满足:$L_1>L_2\Rightarrow C_i(L_1)\geqslant C_i(L_2)$.$L_i$ 表示任务的关键等级,它是一个正整数,最大值为系统中任务的关键级个数.

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.本文的结论可以扩展到任意关键性个数的情况.本文假定任务的参数都是正整数,且运行在单处理器上.下面给出本文中重要的 3 个定义:$hp(\tau_i)$ 表示优先级比任务 τ_i 高的任务集;$hpH(\tau_i)$ 表示优先级比任务 τ_i 高的高关键等级任务集;$hpL(\tau_i)$ 表示优先级比任务 τ_i 高的低关键等级任务集.

本文采用文献[4]中给出的 AMC 调度算法.本文主要研究的问题是计算在 AMC 调度方法下混合关键任务最长的响应时间.所谓任务响应时间是指从任务释放到完成的时间长度.文献[4]给出了两种计算混合关键任务最长的响应时间的方法:AMC-rtb 算法和 AMC-max 算法.AMC-rtb 算法复杂性较低,但是求解的响应时间的值较大,因此将可调度的任务误判为不可调度的数量较多.AMC-max 求解的响应时间值较小,但是算法复杂性较高,随着任务数量和利用率的增加,计算所需时间变得不可接受.本文的目的是寻找一个折中的方法,即任务可调度性比 AMC-rtb 算法高,同时,时间复杂度比 AMC-max 算法低.

下面给出 AMC 调度算法、AMC-rtb 算法和 AMC-max 算法的具体定义.

AMC 调度方法:任务开始运行时以低关键等级时间 $C(LO)$运行,一旦有任务运行时间超过 $C(LO)$,低关键等级的任务就停止运行,高关键等级任务以高关键等级时间 $C(HI)$运行.

高关键等级任务 τ_i 响应时间有 3 种:完全低关键等级下的响应时间 R_i^{LO},完全高关键等级下的响应时间 R_i^{HI} 以及高、低关键等级两者都有时的响应时间 R_i^*.前两者都很简单,分别满足等式(1)和式(2).

$$R_i^{LO}=C_i(LO)+\sum_{\tau_j\in hp(\tau_i)}\left\lceil\frac{R_i^{LO}}{T_j}\right\rceil C_j(LO) \tag{1}$$

$$R_i^{HI}=C_i(HI)+\sum_{\tau_j\in hpH(\tau_i)}\left\lceil\frac{R_i^{HI}}{T_j}\right\rceil C_j(HI) \tag{2}$$

AMC-rtb 算求解 R_i^*:

$$R_i^*=C_i(HI)+\sum_{\tau_j\in hpH(\tau_i)}\left\lceil\frac{R_i^*}{T_j}\right\rceil C_j(HI)+\sum_{\tau_k\in hpL(\tau_i)}\left\lceil\frac{R_i^{LO}}{T_k}\right\rceil C_k(LO) \tag{3}$$

AMC-max 算求解 R_i^*:

$$M(k,s,t)=\min\left\{\left\lceil\frac{t-s-(T_k-D_k)}{T_k}\right\rceil+1,\left\lceil\frac{t}{T_k}\right\rceil\right\} \tag{4}$$

$$R_i^s=C_i(HI)+\sum_{\tau_j\in hpL(\tau_i)}\left(1+\left\lfloor\frac{s}{T_j}\right\rfloor\right)C_j(LO)+\sum_{\tau_k\in hpH(\tau_i)}\left\{\left(M(k,s,R_i^s)C_k(HI)\right)+\left(\left\lceil\frac{t}{T_k}\right\rceil-M(k,s,R_i^s)\right)C_k(LO)\right\} \tag{5}$$

$$R_i^*=\max(R_i^s),\ s\in\left[0,R_i^{LO}\right) \tag{6}$$

2 AMC-PM 算法分析和比较

2.1 AMC-PM算法

算法思路:高关键等级任务 τ_i 只在低关键等级或只在高关键等级下存在临界时刻(critical instant),即与所有优先级比其高的任务一起开始时的响应时间最长,由式(1)和式(2)分别可以求得.困难之处在于如何求关键等级改变时任务的最长响应时间.AMC-rtb 算法假设优先级比其高的任务一直以高关键等级时间运行,由公式(3)可以得到响应时间的上界.AMC-max 算法假设在任务 τ_i 开始运行后的 s 时刻,系统由低关键等级转为高关键等级,亦即任务 τ_i 在前 s 时刻以低关键等级状态运行.由式(1)可知,任务 τ_i 以低关键等级运行的最长时间为 R_i^{LO},一旦运行时间等于 R_i^{LO}, 任务 τ_i 运行结束就不会出现关键等级改变.所以 s 的取值范围为 $\left[0,R_i^{LO}\right)$.

我们并不关心在关键等级改变时任务 τ_i 运行了多长时间,而是关心其自身执行了多少时间.如果任务 τ_i 在低关键等级状态执行了 s 时长时间,则剩余的 $C_i(HI)-s$ 时长是在高关键等级状态执行的.分别求出低关键等级下对应 s 时长时间最长响应时间 $R_i^s(LO)$ 和高关键等级对应 $C_i(HI)-s$ 时长的最长响应时间 $R_i^s(HI)$.$R_i^s(LO)$ 和 $R_i^s(HI)$ 可以分别由式(7)和式(8)求得.

$$R_i^s(LO)=s+\sum_{\tau_j\in hp(\tau_i)}\left(1+\left\lfloor\frac{R_i^s(LO)}{T_j}\right\rfloor\right)C_j(LO) \tag{7}$$

$$R_i^s(HI)=C_i(HI)-s+\sum_{\tau_k\in hpH(\tau_i)}\left\lceil\frac{R_i^s(HI)}{T_k}\right\rceil C_k(HI) \tag{8}$$

在最坏情形下,任务 τ_i 这两部分都是最长响应时间,亦即把它们加起来就是任务响应时间的上界.当 s 遍历 $\left[0,C_i(LO)\right]$ 时,我们取 R_i^s 的最大值即为最长响应时间的上界.由此我们可以得到求解 R_i^* 的 AMC-PM 算法:

$$R_i^s=R_i^s(LO)+R_i^s(HI) \tag{9}$$

$$R_i^*=\max(R_i^s),\ s\in\left[0,C_i(LO)\right) \tag{10}$$

2.2 AMC-PM算法和AMC-rtb算法,AMC-max算法比较

AMC-PM 相对于 AMC-rtb,考虑了比 τ_i 优先级高的高关键等级任务在低关键等级下的运行时间,因此得到

的响应时间要小一些.

AMC-PM 相对于 AMC-max,只需求 $C_i(LO)$ 次运算,而 AMC-max 需要求 R_i^{LO} 次运算.由文献[8,9]可知, $R_i^{LO} \geqslant C_i(LO) \Big/ \left(1-\sum_{j\in hp(i)} U_j\right)$, 所以,AMC-max 算法的时间复杂度至少是 AMC-PM 算法的 $1 \Big/ \left(1-\sum_{j\in hp(i)} U_j\right)$ 倍.随着任务利用率的增加, $C_i(LO)$ 比 R_i^{LO} 小得越来越多,这就意味着 AMC-PM 算法的时间复杂度比 AMC-max 算法越来越低.

下面我们给出一个例子进行具体说明,见表 1.

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	1	2	*HI*	10
τ_2	3	6	*HI*	11
τ_3	4	-	*LO*	12
τ_4	2	4	*HI*	30

对任务 τ_4 分别用 AMC-rtb,AMC-max 和 AMC-PM 求 R_i^* 分别得到 $R_i^* = 40,22,28$. 在 AMC-rtb 算法下, τ_4 被判定为不可调度,而实际上它是可调度的.在 AMC-max 算法和 AMC-PM 算法下, τ_4 的响应时间小于其相对截止时间. $C_4(LO)=2, R_4^{LO}=10$. AMC-max 用的时间是 AMC-PM 的 5 倍左右.

3 实验仿真与结果分析

本节将通过仿真实验来实现本文提出的 AMC-PM 算法,并与文献[4]给出的 AMC-rtb 算法和 AMC-max 算法在随机任务集的可调度性方面进行了比较.

3.1 数据集生成

我们采用与文献[4,6]中类似的生成混合关键性随机任务集的方法,随机生成了两组数据集.第 1 组数据集中最低优先级的高关键等级任务的最长执行时间赋予较小的值(小于 5).第 2 组数据集的参数未做处理.

1. 用文献[10]给出的 UUniFast 算法随机生成一组 n 个任务的利用率 U_i, 其总利用率为 U.
2. 用对数平均分布函数随机生成任务的周期 T_i.
3. 任务的截止期限等于周期 $D_i = T_i$.
4. 任务低关键等级运行时间由上述的利用率和周期生成,即 $C_i(LO) = U_i \cdot T_i$.
5. 任务高关键等级运行时间是低关键等级运行时间倍数,本文设定为 2 倍,即 $C_i(HI) = 2 \cdot C_i(LO)$.
6. 高关键等级任务的数量按一定概率生成,本文设为 0.5.

在我们的实验中,任务集利用率取值范围是 0.03~0.9,步长为 0.03.每一个利用率生成 1 000 组任务集,每组 10 个任务.任务周期的范围是 10~100.

3.2 实验结果分析

图 1 是在第 1 组数据集上进行的实验结果.我们比较了 AMC-rtb,AMC-max 和 AMC-PM 算法在不同任务利用率下的任务的可调度率.可以看出,AMC-PM 算法要优于 AMC-rtb 算法,略低于 AMC-max 算法.特别是当利用率在 0.4~0.7 之间时,AMC-PM 算法相对于 AMC-rtb 算法的优势比较明显,最多可提高 156%.

图 2 是在第 2 组数据集上进行的实验结果.可以看出,当任务的最长执行时间随机生成时,AMC-rtb 算法和 AMC-max 算法要优于 AMC-PM 算法.在这种情形下,我们的方法可调度率比 AMC-rtb 算法和 AMC-max 算法要低.

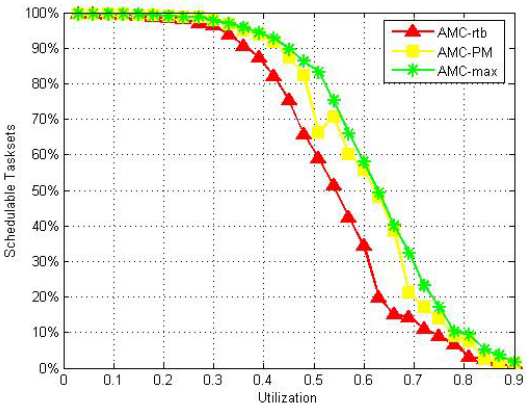

Fig.1 Percentage of schedulable task sets of the first data set

图 1 第 1 组数据上可调度任务集百分比

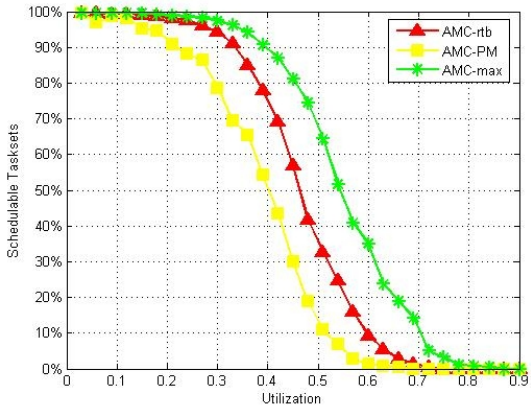

Fig.2 Percentage of schedulable task sets of the second data set

图 2 第 2 组数据上可调度任务集百分比

图 3 也是在第 2 组数据集上进行的实验结果.可以看出,AMC-PM 算法的运行时间要略高于 AMC-rtb 算法的运行时间,但要远低于 AMC-max 算法的运行时间.

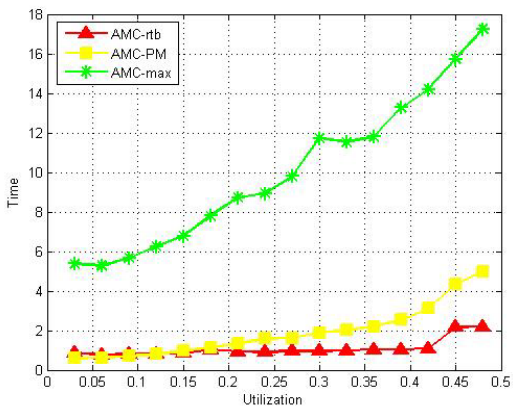

Fig.3 Run time comparison

图 3 运行时间比较

实验结果表明:当任务的最长执行时间较小时,AMC-PM 算法在可调性上优于 AMC-rtb 算法,同时,时间复杂度低于 AMC-max 算法.当任务的最长执行时间随机生成时,AMC-PM 算法在可调性上没有优势.

4 结束语

混合关键系统将不同关键等级的实时应用程序集成到统一的硬件平台上,极大地降低了系统的成本和功耗.混合关键系统自提出后迅速成为近年来实时系统领域的热点问题.本文在文献[4]中提出的混合关键系统 AMC 任务调度算法的基础上给出了一种新的计算混合关键任务响应时间的算法 AMC-PM 算法.该算法时间复杂度比 AMC-max 算法要低,而且在任务最长执行时间较小时,可调性优于 AMC-rtb 算法.

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] 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]

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

[4] 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]

[5] Zhao Q, Gu Z, Zeng H. PT-AMC: Integrating preemption thresholds into mixed-criticality scheduling. In: Proc. of the Conf. on Design, Automation and Test in Europe (DATE). 2013. 141−146. [doi: 10.7873/DATE.2013.042]

[6] Burns A, Davis R. Adaptive mixed criticality scheduling with deferred preemption. In: Proc. of the Real-Time Systems Symp. (RTSS). IEEE, 2014. 21−30. [doi: 10.1109/RTSS.2014.12]

[7] Baruah S, Bonifaci V, D'Angelo G, Li HH, 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]

[8] Bril RJ, Verhaegh WFJ, Pol EJD. Initial values for online response time calculations. In: Proc. of the 15th Euromicro Conf. on Real-Time Systems (ECRTS). IEEE, 2003. 13−22. [doi: 10.1109/EMRTS.2003.1212722]

[9] Davis R, Zabos A, Burns A. Efficient exact schedulability tests for fixed priority real-time systems. IEEE Trans. on Computers, 2008,57(9):1261−1276. [doi: 10.1109/TC.2008.66]

[10] Bini E, Buttazzo GC. Measuring the performance of schedulability tests. Real-Time Systems, 2005,30(1-2):129−154. [doi: 10.1007/s11241-005-0507-9]

白恩慈(1988－),男,河北邯郸人,博士生,CCF 学生会员,主要研究领域为实时系统.

张伟哲(1976－),男,博士,教授,博士生导师,CCF 专业会员,主要研究领域为高性能计算,并行与分布式系统,嵌入式系统,计算机网络安全.