·开发研究与工程应用· 文章编号:1000-3428(2016)05-0288-04 文献标识码:A 中图分类号:TP316.2

单处理器平台下的严格周期任务可调度性判定

陈进朝,杜承烈

(西北工业大学计算机学院,西安 710129)

摘 要: 针对系统调度理论研究中的严格周期任务可调度性判定问题,从研究实时系统中严格周期任务之间的相互干涉关系出发,提出一种基于特征任务的可调度性判定方法。分析严格周期任务在单一处理器平台下无冲突运行的时间约束,计算任务所能使用的全部空余时间,进而界定连续空余时间是否满足任务执行的需要,得到一个可调度性判定的充要条件。实验结果表明,与特征映射任务分配方法相比,该方法能够减少判定时间,提高判定成功率,具有更优的可调度判定性能。

关键词: 严格周期任务;可调度性判定;单处理器;特征任务;实时系统;空余时间

中文引用格式: 陈进朝,杜承烈. 单处理器平台下的严格周期任务可调度性判定[J]. 计算机工程,2016,42(5):288-291.
英文引用格式: Chen Jinchao, Du Chenglie. Schedulability Test for Strictly Periodic Tasks on Uniprocessor Platform[J]. Computer Engineering, 2016, 42(5):288-291.

Schedulability Test for Strictly Periodic Tasks on Uniprocessor Platform

CHEN Jinchao, DU Chenglie

(School of Computer Science, Northwestern Polytechnical University, Xi'an 710129, China)

【Abstract】 Aiming at the problem of schedulability test for strictly periodic tasks in theoretical study of system scheduling, this paper researches the interference relationship between strictly periodic tasks in a real-time system, and presents an eigentask-based schedulability test method. It analyzes the time constraints of strictly periodic tasks running on a uniprocessor platform, calculates all free time, and provides a sufficient and necessary schedulability test condition by determining whether enough continuous free time is available for the task's execution. Experimental results show that, compared with Eigen-mapping Task Assignment(EMTA) method, the proposed method can reduce the time consumption required for the test and improve the success ratio. It has better schedulability test performance.

【Key words】 strictly periodic task; schedulability test; uniprocessor; eigentask; real-time system; free time
DOI: 10.3969/j.issn.1000-3428.2016.05.050

1 概述

严格周期任务因其具有的可预测性[1]和易实现性[2],被广泛应用于高安全实时系统中[3]。为了保障这些系统的正确性与可靠性,严格周期任务的可调度性判定成为实时系统调度理论研究的一个核心问题[4-6]。然而,该问题已经被证明是一个 NP 完全问题[7],目前仅有多项式时间内的近似解决方案[8],并且需要花费大量的计算时间。本文引入文献[9]中提出的特征任务(eigentask),从任务之间的相互干涉关系入手,分析严格周期任务在单一处理器下可调度的时间约束,计算任务所能使用的全部空余时间,并由此得到一个更有效的可调度性判定充要条件。

2 系统模型及相关定义

假设 $T=\{\tau_1,\tau_2,\cdots,\tau_n\}$ 是一个含有 n 个严格周期任务的集合,集合中的任务相互独立且不允许抢占。每个任务用三元组 $\tau_i=\langle c_i,p_i,s_i\rangle$ $(i\in[1,n])$ 来表示,其中,c_i,p_i,s_i 分别表示任务 τ_i 的执行时间、周期和开始时间,且满足关系 $0<c_i\leqslant p_i$ 和 $0\leqslant s_i\leqslant p_i-c_i$。如果任务的开始时间尚未分配,则采用二元组 $\langle c_i,p_i\rangle$ 来表示任务 τ_i。本文假定任务的时间参数全部为整数,即 $c_i,p_i,s_i\in\mathbb{N}$。任务利用率 u_i 定义为任务执行时间与周期的比例,即 $u_i=c_i/p_i$。

每个任务均由无穷多个作业(job)构成。在严格周期的约束下,同一个任务的 2 个连续作业之间

基金项目: 国家自然科学基金资助项目(61373120)。
作者简介: 陈进朝(1987-),男,博士研究生,主研方向为分布式实时仿真系统;杜承烈,教授、博士生导师。
收稿日期: 2015-09-28 **修回日期:** 2015-10-22 **E-mail:** cjz3809@163.com

的时间间隔是固定的,且等于任务的周期[10],因此,对于任务 τ_i,其第 k 个作业的开始运行时间为 $s_i + kp_i$,结束时间为 $s_i + kp_i + c_i$,即该作业运行于时间区间$[s_i + kp_i, s_i + kp_i + c_i)$。令 $B_i^k(s_i)$ 表示任务 τ_i 第 k 个作业所使用的时间区间,则 $B_i^k(s_i) = [s_i + kp_i, s_i + kp_i + c_i)$。同时,用 E_i 表示任务 τ_i 所有工作所占用的时间,即 $E_i = \bigcup_{k\in\mathbb{N}} B_i^k(s_i)$。本文使用的周期任务模型如图 1 所示。

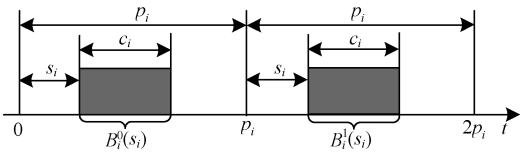

图 1　严格周期任务模型

3　可调度性分析

本文在严格周期任务模型的基础上,分析 2 个任务之间的可调度约束,并利用特征任务计算出当前可用的空余时间,从而得到多任务情形下的可调度性判定条件。

3.1　2 个任务的可调度性分析

依据严格周期的约束,对于任务 τ_i,其第 k 个工作所占用的时间区间为:

$$B_i^k(s_i) = [s_i + kp_i, s_i + kp_i + c_i) \tag{1}$$

任务 τ_i 和 τ_j 是可调度的,当且仅当它们的所有工作在时间上没有任何冲突[11],即任务 τ_i 和 τ_j 是可调度的,当且仅当:

$$\forall k,l \geqslant 0, B_i^k(s_i) \cap B_j^l(s_j) = \phi \tag{2}$$

尽管式(2)是 2 个任务可调度性判定的充要条件,然而由于任务的工作是无限产生的,不能定量地计算出任务全部工作所占用的时间单元,因此该条件不能被直接使用。文献[12]提出了一个更高效方便的判定条件。

定理 1　任务 $\tau_i = \langle c_i, p_i, s_i\rangle$ 和 $\tau_j = \langle c_j, p_j, s_j\rangle$ 是可调度的,当且仅当:

$$c_i \leqslant (s_j - s_i) \bmod (g_{i,j}) \leqslant g_{i,j} - c_j \tag{3}$$

其中,$g_{i,j}$是任务 τ_i 和 τ_j 周期的最大公约数,即 $g_{i,j} = GCD(p_i, p_j)$。

式(3)是研究严格周期任务可调度性判定的基础,但在多任务情形下,它仅是一个充分条件[2],并不能完全确定一个多任务集合的可调度性。本文基于文献[9]提出的特征任务来解决这一问题。

对于每个任务 $\tau_r = \langle c_r, p_r\rangle$ 都存在一个特殊的任务 $\tau_{r'}$,其周期与原任务的周期相同,但执行时间却是单位时间 1,即 $\tau_{r'} = \langle 1, p_r\rangle$。$\tau_{r'}$被视为任务 τ_r 的特征任务。基于特征任务的可调度性分析主要有以下 2 个步骤:

(1)利用原任务的特征任务,求出特征任务在可调度时的所有可用空余时间。

(2)计算特征任务可用空余时间的最大长度,如果该长度不小于原任务的执行时间,则该任务是可调度的。

基于上述分析方法,本文首先求出特征任务的可用空余时间,进而给出原任务可调度性判定的充要条件。

3.2　特征任务可用空余时间的计算

本文采用取模操作[13]来计算特征任务所有的可用空余时间。假设 $T = \{\tau_1, \tau_2, \cdots, \tau_n\}$ 是系统中已存在任务的集合,$\tau_r = \langle c_r, p_r\rangle$ 是需要进行可调度性判定的任务。依据特征任务的定义,任务 $\tau_{r'} = \langle 1, p_r\rangle$ 是 τ_r 的特征任务。

对集合 T 中每一个任务 τ_i 所占用的时间在周期 p_r上进行取模,得到 τ_i 对 $\tau_{r'}$的干涉时间,即 $\tau_{r'}$所不能使用的时间。如果 x 是任务 τ_i 所占用的一个时间单元,那么 $x' = x \bmod p_r$ 即 τ_i 对 $\tau_{r'}$的干涉时间单元。以例 1 说明任务所占用的时间与干涉时间之间的关系。

例 1　考虑一个含有 2 个任务的集合 $T = \{\tau_1, \tau_2\}$,其中,$\tau_1 = \langle 1,4,0\rangle$;$\tau_2 = \langle 1,12,1\rangle$;$\tau_3 = \langle 1,8\rangle$,是一个特征任务。执行时间与干涉时间如图 2 所示。该图上方的坐标轴显示了任务 τ_1 和 τ_2 在时间区间$[0,24)$上的执行序列,下方的圆环表示对 τ_1 和 τ_2 所占用的时间在 τ_3 的周期上进行取模,得到 τ_1 和 τ_2 对 τ_3 的干涉时间。可以看出,τ_1 和 τ_2 所占用的每一个时间单元都有一个对应的干涉时间单元。τ_1, τ_2 对 τ_3 的干涉时间分别为$\{0,4\}$和$\{1,5\}$。

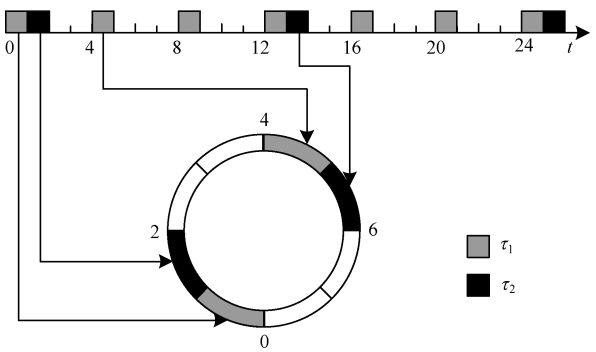

图 2　执行时间与干涉时间

令 $ITU(i,r)$表示任务 τ_i 对特征任务 $\tau_{r'}$的干涉时间,则:

$$ITU(i,r) = \{x \mid x = (e) \bmod (p_r), \forall e \in E_i\} \tag{4}$$

利用干涉时间可以得到特征任务与单一任务是否可调度的判定条件。

定理 2　特征任务 $\tau_{r'} = \langle 1, p_r, s\rangle$ 与 $\tau_i = \langle c_i, p_i, s_i\rangle$ 是可调度的,当且仅当:

$$s \in Z(r) \backslash ITU(i,r) \tag{5}$$

其中,$Z(r)=[0,p_r-1]$。

证明:依据严格周期任务模型,任务的开始时间不能超过任务的周期,即 $s\in[0,p_r-1]$。同时,2 个任务是可调度的,当且仅当它们的所有工作没有时间冲突,可以表示为 $E_{r'}\cap E_i=\phi$。

$$\begin{aligned}
&E_{r'}\cap E_i=\phi\\
&\Leftrightarrow \bigcup_{k\in\mathbb{N}} B_{r'}^k(s)\cap E_i=\phi\\
&\Leftrightarrow \bigcup_{k\in\mathbb{N}} [s+kp_r,s+kp_r+1)\cap E_i=\phi\\
&\Leftrightarrow (s+kp_r)\notin E_i,\forall k\in\mathbb{N}\\
&\Leftrightarrow s\notin ITU(i,r)
\end{aligned}$$

由此可知,$s\in Z(r)\backslash ITU(i,r)$。此定理得证。

定理 2 给出了特征任务 $\tau_{r'}$ 与单一任务 τ_i 的可调度性判定条件。基于此条件,把 τ_i 推广到集合 T 中的所有任务,得到多任务情形下 $\tau_{r'}$ 可调度的判定依据。

定理 3 特征任务 $\tau_{r'}=\langle 1,p_r,s\rangle$ 与集合 T 中的所有任务是可调度的,当且仅当:

$$s\in Z(r)\backslash \bigcup_{\tau_i\in T} ITU(i,r) \tag{6}$$

证明:由定理 2 可知,$\tau_{r'}$ 和集合 T 中的任一任务 τ_i 是可调度的,当且仅当式(5)成立。若 $\tau_{r'}$ 和集合 T 的所有任务是可调度的,则 $\forall\tau_i\in T, s\in Z(r)\backslash ITU(i,r)$,根据集合运算,该式等价于 $s\in Z(r)\backslash\bigcup_{\tau_i\in T} ITU(i,r)$。此定理得证。

由定理 3 可知,当且仅当 $\tau_{r'}$ 的开始时间是集合 $Z(r)\backslash\bigcup_{\tau_i\in T} ITU(i,r)$ 中的一个整数时,$\tau_{r'}$ 和集合 T 中的所有任务是可调度的,这意味着 $Z(r)\backslash\bigcup_{\tau_i\in T} ITU(i,r)$ 包含了所有能够使 $\tau_{r'}$ 在当前多任务情形下可调度的空余时间。令 $F_T(r)$ 表示这些可用的空余时间,则:

$$F_T(r)=Z(r)\backslash\bigcup_{\tau_i\in T} ITU(i,r) \tag{7}$$

3.3 可调度性判定

依据特征任务可用的空余时间可以导出一个具有更广适用范围的可调度性判定条件。该判定条件需要使用文献[9]中提出的函数 $LLC(S)$ 来表示一个非空集合 S 中连续整数的最长长度。

定理 4 任务 $\tau_r=\langle c_r,p_r\rangle$ 与集合 T 中所有任务是可调度的,当且仅当:

$$c_r\leqslant LLC(F_T(r)) \tag{8}$$

证明:先进行充分性证明。如果 $c_r\leqslant LLC(F_T(r))$,那么一定存在一个整数 s,使得从 s 到 $s+c_r-1$ 连续 c_r 个整数都在集合 $F_T(r)$ 中。依据定理 3 可知,当特征任务 $\tau_{r'}=\langle 1,p_r\rangle$ 的开始时间是区间 $[s,s+c_r-1]$ 中的任意一个整数时,$\tau_{r'}$ 与集合 T 中所有任务是可调度的,即 $\forall\tau_i\in T,\forall k,l\in\mathbb{N}$,并且:

$$\begin{aligned}
&B_{r'}^l(s+m)\cap B_i^k(s_i)=\phi,\forall m\in[0,c_r-1]\\
&\Rightarrow(\bigcup_{m=0}^{c_r-1} B_{r'}^l(s+m))\cap B_i^k(s_i)=\phi\\
&\Rightarrow(\bigcup_{m=0}^{c_r-1}[s+lp_r+m,s+lp_r+m+1))\cap B_i^k(s_i)=\phi\\
&\Rightarrow[s+lp_r,s+lp_r+c_r)\cap B_i^k(s_i)=\phi\\
&\Rightarrow B_r^l(s)\cap B_i^k(s_i)=\phi
\end{aligned}$$

因此,任务 τ_r 与集合 T 中的所有任务是可调度的,充分性成立。

下面采用反证法来证明式(8)是必要的。如果式(8)不成立,即 $c_r>LLC(F_T(r))$ 时,τ_r 至少与集合 T 的一个任务有时间上的冲突。

当 $c_r>LLC(F_T(r))$ 时,必然存在一个整数 s,使得 s 被任务 τ_r 的第一个工作所占用,同时不在集合 $F_T(r)$ 中,即 $s\in[s_r,s_r+c_r)$ 且 $s\notin F_T(r)$。由定理 3 可知,$s\notin F_T(r)$ 表明特征任务 $\tau_{r'}=\langle 1,p_r,s\rangle$ 至少与集合 T 中的一个任务冲突,即 $\exists\tau_i\in T,\exists k,l\in\mathbb{N}, B_{r'}^k(s)\cap B_i^l(s_i)\neq\phi$。现考虑 τ_r 的第 k 个工作与 τ_i 的第 l 个工作:

$$\begin{aligned}
B_r^k(s_r)\cap B_i^l(s_i)&=[s_r+kp_r,s_r+kp_r+c_r)\cap B_i^l(s_i)\\
&\supseteq[s+kp_r,s+kp_r+1)\cap B_i^l(s_i)\\
&=B_{r'}^k(s)\cap B_i^l(s_i)\neq\phi
\end{aligned}$$

由此可知任务 τ_r 与 τ_i 是不可调度的,这证明了式(8)的必要性。此定理得证。

定理 4 给出了一个判断任务是否可调度的充要条件,它不需要任何形式的迭代操作,适用于多任务集合,可应用于严格周期任务系统的设计与验证。

4 实验与结果分析

通过与文献[9]提出的特征映射任务分配(Eigen-mapping Task Assignment Algorithm,EMTA)方法进行对比,从时间消耗和成功率 2 个方面评估本文方法的性能。时间消耗是指不同方法对任务做出是否可调度判定所花费的时间,可以反映出方法的执行速度和效率。成功率是指判定为可调度的任务数量与总任务数量的比例,体现出判定方法的可靠性与稳定性。

随机任务集的产生过程与文献[14-15]中使用的一致:任务的周期从集合 $\{1\ 500\times 2^x: x\in[0,4]\}$ 中随机选择,任务的利用率服从期望为 0.2 的指数分布,任务的执行时间由周期与利用率合成,即 $c_i=p_iu_i$。

图 3 给出了 EMTA 及本文方法在不同任务数目下的平均判定时间。可以看出,2 种方法所需的判定时间都随任务数目的增大而增加,而在任务数目相同的情况下,本文方法所花费的判定时间更少,具有更快的速度和更高的效率。

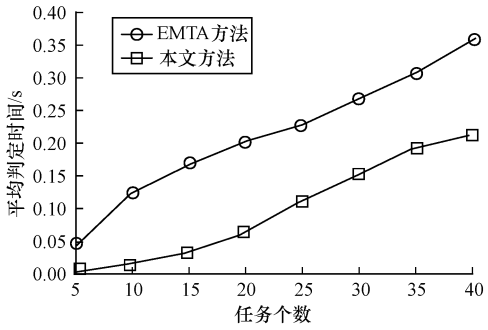

图 3　平均判定时间随任务数目的变化

图 4 对 2 种方法的判定成功率进行了对比。可以看出,在任务数目相同的情况下,本文方法比 EMTA 方法具有更高的判定成功率,这是因为本文提出的判定条件是充要条件,更多数量的任务能够被判定为是可调度的。

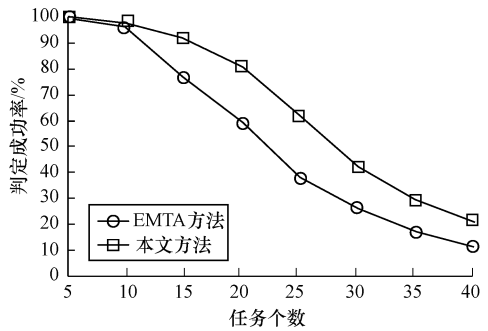

图 4　判定成功率随任务数目的变化

上述实验结果表明,相对 EMTA 方法而言,本文方法在严格周期任务可调度性判定上表现出更优的性能,不仅有更快的判定速度,而且具有更高的判定成功率与可靠性。

5　结束语

本文对严格周期任务在单处理器下的可调度性进行研究,分析任务可调度时的时间约束,给出并证明多任务情况下的可调度性判定条件。实验结果表明,本文判定方法具有较快的判定速度与较高的判定成功率,可用于严格周期任务系统的设计。下一步工作将针对具有通信约束的周期任务进行可调度性研究,并验证本文方法能否适用于新系统的设计。

参考文献

[1] Jeffay K, Stanat D, Martel C. On Non-preemptive Scheduling of Period and Sporadic Tasks [C]// Proceedings of the 12th Real-time Systems Symposium. Washington D. C., USA: IEEE Computer Society, 1991: 129-139.

[2] Kermia O, Sorel Y. Schedulability Analysis for Non-preemptive Tasks Under Strict Periodicity Constraints[C]// Proceedings of the 14th IEEE International Conference on Embedded and Real-time Computing Systems and Applications. Washington D. C., USA: IEEE Press, 2008: 25-32.

[3] 张永悦,孙　瑜,李　允,等. 复杂实时系统可调度性判定工具的研究与实现[J]. 计算机工程,2013,39(1):270-274.

[4] 毛羽刚,张拥军,金士尧,等. 一种改进的分布强实时系统可调度性分析算法[J]. 软件学报,2001,12(2):298-302.

[5] 王永吉,陈秋萍. 单调速率及其扩展算法的可调度性判定[J]. 软件学报,2004,15(6):799-814.

[6] 徐建华,李　允. 基于单调速率的可调度性判定改进算法[J]. 计算机工程,2011,37(22):45-47.

[7] Korst J. Periodic Multiprocessor Scheduling[D]. Eindhoven, the Netherlands: Eindhoven University of Tech-nology, 1991.

[8] Baruah S, Chakraborty S. Schedulability Analysis of Non-preemptive Recurring Real-time Tasks[C]// Proceedings of the 20th International Parallel and Distributed Processing Symposium. Washington D. C., USA: IEEE Computer Society, 2006.

[9] Chen Jinchao, Du Chenglie, Xie Fei, et al. Schedulability Analysis of Non-preemptive Strictly Periodic Tasks in Multi-core Real-time Systems[J]. Real-Time Systems, 2015, 51(1): 1-33.

[10] Marouf M, Sorel Y. Scheduling Analysis for Non-preemptive Hard Real-time Tasks with Strict Period[C]// Proceedings of the 16th International Conference on Emerging Technologies and Factory Automation. Washington D. C., USA: IEEE Press, 2011: 1-8.

[11] Al-Sheikh A, Brun O, Hladik P E, et al. Strictly Periodic Scheduling in IMA-based Architectures[J]. Real-Time Systems, 2012, 48(4): 359-386.

[12] Korst J, Aarts E, Lenstra J, et al. Periodic Multiprocessor Scheduling[M]//Aarts E H L, van Leeuwen J, Rem M. PARLE'91 Parallel Architectures and Languages Europe. Berlin, Germany: Springer-Verlag, 1991.

[13] Kunth D E. The Art of Computer Programming [M]. New York, USA: Addison-Wesley Educational Publishers Inc., 1973.

[14] Pira C, Artigues C. Line Search Method for Solving a Non-preemptive Strictly Periodic Scheduling Problem [J]. Journal of Scheduling, 2014, 31(6): 1-17.

[15] Al-Sheikh A, Brun O, Hladik P E, et al. A Best-response Algorithm for Multiprocessor Periodic Scheduling[C]// Proceedings of the 23rd Euromicro Conference on Real-time Systems. Washington D. C., USA: IEEE Press, 2011: 228-237.

编辑　金胡考