·体系结构与软件技术· 文章编号:1000-3428(2017)12-0065-04 文献标志码:A 中图分类号:TP316.2

全局固定优先级实时调度算法分析

梁 浩[a],晏 立[b],沈项军[b]

(江苏大学 a. 研究生院; b. 计算机科学与通信工程学院,江苏 镇江 212013)

摘 要: 基于截止期分析和响应时间分析可以对全局固定优先级实时调度算法进行可调度性判定。而传统方法在实时任务中带入作业,处理器无法满足实时任务的计算需求。为此,提出一种可调度性判定方法。通过区分实时任务在具有和没有带入作业时产生的干涉,考虑带入作业的个数与实时系统中处理器的个数相关。实验结果表明,该方法能够减少计算的干涉量,得到一个更紧密的可调度性判定条件,提高多处理器实时系统中通过可调度性判定的任务数量。

关键词: 实时系统;多处理器;全局调度;可调度性判定;干涉

中文引用格式: 梁 浩,晏 立,沈项军. 全局固定优先级实时调度算法分析[J]. 计算机工程,2017,43(12):65-68.

英文引用格式: LIANG Hao, YAN Li, SHEN Xiangjun. Analysis of Real-time Scheduling Algorithm with Global Fixed Priority[J]. Computer Engineering, 2017, 43(12):65-68.

Analysis of Real-time Scheduling Algorithm with Global Fixed Priority

LIANG Hao[a], YAN Li[b], SHEN Xiangjun[b]

(a. Graduate School; b. School of Computer Science and Telecommunnication Engineering, Jiangsu University, Zhenjiang, Jiangsu 212013, China)

【Abstract】 The schedulability can be determined based on deadline analysis and response time analysis for the global Fixed Priority(FP) real-time scheduling algorithm. The traditional method takes into account the real-time tasks with a carry-in job. The processor cannot meet the computing requirement of real-time tasks. So this paper proposes a schedulability method. The interference caused by the real time task is analyzed, which takes into account the number of jobs and the number of processors in real-time system. Experimental results show that the method can reduce the amount of interference, obtain a more compact schedulability criterion, and increases the number of tasks that can be passed by schedulability in multiprocessor real-time system.

【Key words】 real-time system; multiprocessor; global scheduling; schedulability determination; interference

DOI: 10.3969/j.issn.1000-3428.2017.12.012

0 概述

固定优先级(Fixed Priority, FP)实时调度算法能够保证较低的时间复杂度,广泛应用于实时系统中[1-3]。FP 算法实现简单,与 EDF、EDZL、LLF 等动态调度算法相比,不需要在运行时改变任务的优先级,大大减少了运行开销。在多处理器实时系统中,全局 FP 算法限制了系统中任务抢占和迁移的数目,使系统开销大幅减少,能得到较高的执行效率[4]。由于 FP 算法的简单性,在物联网等小型实时系统中得到大量应用,降低了系统费用,提高了运行效率,有很大的经济意义与应用价值。

在实时系统设计中,必须进行可调度性判定[5]。可调度性判定决定了实时系统能否正确地调度所有实时任务,以及系统是否能可靠运行,否则实时系统必须重新设计[6-8]。

文献[9]通过对实时任务截止期的分析进行了全局 FP 算法的可调度性判定,又通过响应时间分析进行了可调度性判定[10]。实时任务中的带入作业,对可调度性判定存在很大的影响[11-13]。在文献[9]的方法中,假定了实时系统中的所有任务都存在带入作业,给出的结果不够理想。实际上,实时任务不一定都存在带入作业,本文考虑实时任务具有带入作业的条件,对不存在带入作业的任务进行区分,给出一个更紧密的可调度性判定方法。

1 系统模型

本文的系统模型是一个多处理器实时系统,系

基金项目: 国家自然科学基金(61572240)。

作者简介: 梁 浩(1972—),男,实验师、硕士,主研方向为实时系统;晏 立,教授;沈项军,教授、博士。

收稿日期: 2017-02-16 **修回日期:** 2017-04-05 **E-mail:** hliang@ujs.edu.cn

统中有 m 个相同处理器。运行实时任务集合 $\tau=\{\tau_1,\tau_2,\cdots,\tau_n\}$，任务为实时零星任务。实时任务用三元组 $\tau_i=(C_i,D_i,T_i)$ 描述，其中，τ_i 表示实时任务，C_i 为最坏执行时间，D_i 为相对截止期，T_i 为最小到达时间(即周期)。在本文的模型中，任务 τ_i 的作业为 $j_i=\{j_{i,1},j_{i,2},\cdots,j_{i,k},\cdots\}$。任务的最小间隔时间为 T_i，T_i 也称任务的执行周期。本文假定任务到达后立即释放，等待调度。处理器空闲时，就绪队列中的任务立即得到执行。

在系统中，作业是可迁移的，迁移的开销忽略不计，或包含在 C_i 中。

定义 1 FP 调度算法。FP 调度算法在系统设计时确定优先级，运行期间不变化。

RM 算法和 DM 算法是最常用的 2 种 FP 算法，本文讨论的方法同时适用于这 2 种算法。

定义 2 可调度性判定。在确定的调度算法下，判断实时系统中所有任务是否都能被正确调度，称为可调度性判定。

定义 3 负载。任务 τ_i 在区间 $[a,b)$ 的负载 $W_i(a,b)$ 定义为在某种调度策略下，任务 τ_i 在区间 $[a,b)$ 的执行时间总和。设区间 $[a,b)$ 的长度为 L，$W_i(a,b)$ 也可记作 $W_i(L)$。

定义 4 干涉。在任务 τ_k 执行时，被其他任务 τ_i 抢占了正在执行的处理器，称为受到了干涉。在长度为 L 的时间区间中，$I_K(L)$ 表示任务 τ_κ 受到的总干涉；而 $I_{i,k}(L)$ 则表示某一个任务 τ_i 对任务 τ_κ 的干涉，它们之间满足 $I_k(L)\geqslant I_{i,k}(L)$。

任务 τ_κ 执行时，处理器被其他任务 τ_i 过多地抢占，以致错过截止期。通过计算其他任务 τ_i 产生的干涉量，可以检查任务 τ_κ 是否产生超时。

任务 τ_κ 就绪后，由于优先级低于 τ_i 不能执行，形成干涉子区间，这些干涉子区间的累积构成的总长度为 τ_κ 受到的总干涉。总干涉难于计算，目前都是计算其上界。

定义 5 带入作业。在一个作业执行的时间窗口 $[a,b)$ 中，如果该作业在 a 之前释放，并且在时间 a 还未完成，称为带入作业。

2 FP 算法可调度性分析

图 1 是任务 τ_κ 的一个调度区间 $[t_a,t_d)$，长度为 L，τ_κ 的作业 j_κ 在这个区间执行。τ_κ 的相对截止期为 D_κ，在时刻 t_a，j_κ 到达并释放，截止时刻为 t_d。

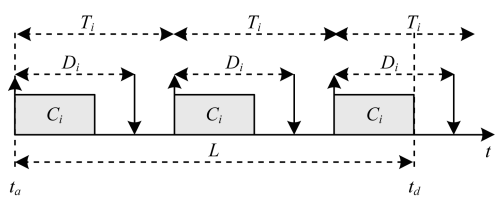

图 1 任务 τ_i(无带入作业)对 τ_κ 的干涉

在图 1 中，任务 τ_i 的负载上界为：

$$W_i^{\mathrm{NC}}(L)=N_i^{\mathrm{NC}}(L)C_i+\min(C_i,L-N_i^{\mathrm{NC}}(L)T_i) \tag{1}$$

其中：

$$N_i^{\mathrm{NC}}(L)=\left\lfloor\frac{L}{T_i}\right\rfloor \tag{2}$$

符号 $\lfloor\ \rfloor$ 表示向下取整。

在长度为 L 的区间中，如果任务 τ_κ 是可调度的，优先级高于 τ_k 的任务 τ_i 对 τ_κ 产生干涉的上界为：

$$I_{i,k}^{\mathrm{NC}}(L)=\min(W_i^{\mathrm{NC}}(L),L-C_k+1) \tag{3}$$

在图 1 中，任务 τ_i 没有考虑存在带入作业(用上标 NC 表示)。在实际运行的系统中，由于任务执行的交错性，一般会存在带入作业。任务 τ_i 有带入作业的情况如图 2 所示。

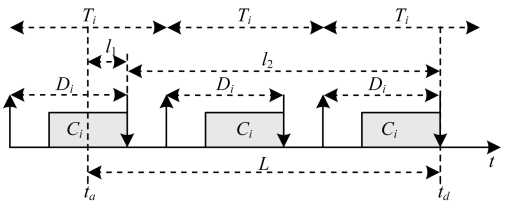

图 2 存在带入作业时 τ_i 对 τ_κ 的干涉

图 2 表示了有带入作业的任务 τ_i 对其在区间 $[t_a,t_d)$ 中最大负载情况对应的场景。

具有带入作业的高优先级任务 τ_i 在这个区间的负载上界(用上标 D 表示)为：

$$W_i^{\mathrm{D}}(L)=N_i^{\mathrm{D}}(L)C_i+\min(C_i,L+D_i-C_i-N_i^{\mathrm{D}}(L)T_i) \tag{4}$$

其中：

$$N_i^{\mathrm{D}}(L)=\left\lfloor\frac{L+D_i-C_i}{T_i}\right\rfloor \tag{5}$$

如果任务 τ_κ 是可调度的，任务 τ_i 干涉的上界为：

$$I_{i,k}^{\mathrm{D}}(L)=\min(W_i^{\mathrm{D}}(L),L-C_k+1) \tag{6}$$

文献[9]根据对任务 τ_κ 相对截止期的分析，考虑了所有高优先级任务 τ_i 产生的干涉和系统中处理器个数，给出了 FP 算法的可调度性判定的公式：

$$\sum_{\forall i\in hp(k)}I_{i,k}^{\mathrm{D}}(D_k)<m(D_k-C_k+1) \tag{7}$$

其中，$\forall i\in hp(k)$ 表示所有优先级高于 τ_κ 的任务 τ_i，m 是实时系统中处理器的个数。

由此可以推出，如果任务集中的每一个任务 τ_κ 满足不等式(8)，一个周期(零星)任务集是可调度的：

$$D_k\geqslant C_k+\left\lfloor\frac{1}{m}\left(\sum_{\forall i\in hp(k)}I_{i,k}^{\mathrm{D}}(D_k)\right)\right\rfloor \tag{8}$$

在式(7)中，所有产生干涉的任务都认为是具有带入作业的。然而，在实际的系统中，并非所有任务在运行时都有带入作业。因为 $D_i\geqslant C_i$，否则任务 τ_i 还未完成就会错过截止期，比较式(1)与式(4)，可知 $W_i^{\mathrm{D}}(D_k)\geqslant W_i^{\mathrm{NC}}(D_k)$，即 $I_{i,k}^{\mathrm{D}}(D_k)\geqslant I_{i,k}^{\mathrm{NC}}(D_k)$。如

果找出无带入作业的任务,对无带入作业的任务,用 $I_{i,k}^{NC}(L)$ 替换式(6)中的 $I_{i,k}^{D}(D_k)$,则会减少式(8)右边的值,使式(8)的可调度判定的条件更加宽松。

定义 6　最大连续忙区间[14]。如图 3 所示,设作业 j_κ 在 t_a 释放,其截止期为 t_d。设 $t_0 \leqslant t_a$,在区间 $[t_0, t_a)$,系统中处理器都在执行作业,直至时刻 t_0 才出现空闲处理器。因为在区间 $[t_0, t_a)$ 系统都在忙碌,所以该区间称为最大连续忙区间,其长度为 $A_k = t_a - t_0$。

最大连续忙区间如图 3 所示。

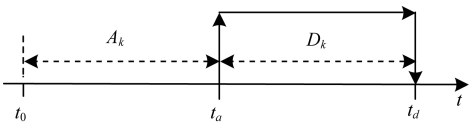

图 3　最大连续忙区间

因为系统有 m 个处理器,在最大连续忙区间中,最多只可能有 m 个任务在执行。如果任务 τ_κ 是可调度的,τ_κ 在 t_a 释放且能够执行,说明在任务 τ_κ 开始执行时,至多只有 $m-1$ 个其他任务 τ_i 的作业执行。也就是说,系统中至多只可能有 $m-1$ 个任务 τ_i 存在带入作业。

下面选择存在带入作业的 $m-1$ 个任务。

式(9)表示任务 τ_i 是否存在带入作业在长度为 D_κ 的区间中对任务 τ_κ 产生干涉的时间差:

$$I_{i,k}^{DIFF}(D_k) = I_{i,k}^{D}(D_k) - I_{i,k}^{NC}(D_k) \tag{9}$$

取 $m-1$ 个最大的 $I_{i,k}^{DIFF}(D_k)$,则得到任务 τ_i 存在带入作业时增加干涉的上界。这样就可以将所有高优先级任务 τ_i 按照无带入作业计算对 τ_κ 产生的干涉,再加上 $m-1$ 个最大的 $I_{i,k}^{DIFF}(D_k)$,则可得到任务 τ_i 对任务 τ_κ 产生的干涉上界:

$$\sum_{\forall i \in hp(k)} I_{i,k}^{NC}(D_k) + \sum_{(m-1)\text{个最大的}} I_{i,k}^{DIFF}(D_k) \tag{10}$$

由此,可得到改进的 FP 算法的可调度性判定方法。

如果任务集中的每一个任务 τ_κ 满足不等式(10),一个周期(零星)任务集是可调度的:

$$D_k \geqslant C_k + \left\lfloor \frac{1}{m}\left(\sum_{\forall i \in hp(k)} I_i^{NC}(D_k) + \sum_{(m-1)\text{个最大的}} I_{i,k}^{DIFF}(D_k) \right) \right\rfloor \tag{11}$$

文献[15]对全局 FP 调度进行了响应时间分析,给出了如下定理。

定理 2　在多处理器系统中用 FP 算法调度,任务 τ_κ 响应时间的上界可以以 $R_k^{UB} = C_k$ 为初始值,用下列表达式对 R_k^{UB} 进行不动点迭代:

$$R_k^{UB} \leftarrow C_k + \left\lfloor \frac{1}{m} \sum_{\forall i \in hp(k)} I_{i,k}^{R}(R_k^{UB}) \right\rfloor \tag{12}$$

其中,$I_{i,k}^{R}(R_k^{UB}) = \min(I_{i,k}^{D}(D_k), R_k^{UB} - C_k + 1)$。

迭代的初始值 $R_k^{UB} = C_k$。在迭代中,如果 $R_k^{UB} > D_k$,说明任务 τ_κ 不可调度,即实时系统是不可调度的;如果 R_k^{UB} 收敛,R_k^{UB} 就是任务 τ_κ 的响应时间上界。

如果 R_k^{UB} 收敛,可以看出,$C_k \leqslant R_k^{UB} \leqslant D_k$。比较式(8)和式(12),可以看出,式(8)是对任务 τ_κ 的截止期进行分析,式(12)对任务 τ_κ 的响应时间上界进行分析。两者都是利用高优先级任务 τ_i 对 τ_κ 的干涉进行可调度性判定。

式(12)与式(8)一样,也是认为所有任务都存在带入作业。将式(9)的方法用于式(12),得到:

定理 3　在多处理器系统中用 FP 算法调度,任务 τ_κ 响应时间的上界可以以 $R_k^{UB} = C_k$ 为初始值,用下列表达式对 R_k^{UB} 进行不动点迭代:

$$R_k^{UB} \leftarrow C_k + \left\lfloor \frac{1}{m}\left(\sum_{\forall i \in hp(k)} I_i^{NC}(R_k^{UB}) + \sum_{(m-1)\text{个最大的}} I_{i,k}^{DIFF}(R_k^{UB}) \right) \right\rfloor \tag{13}$$

其中,$I_{i,k}^{DIFF}(R_k^{UB}) = I_{i,k}^{D}(R_k^{UB}) - I_{i,k}^{NC}(R_k^{UB})$。

迭代初始值 $R_k^{UB} = C_k$。在迭代中,如果 $R_k^{UB} > D_k$,说明任务 τ_κ 不可调度,即实时系统是不可调度的;如果对于所有的任务 τ_κ,R_k^{UB} 均收敛,实时系统是可调度的。

定理 3 的证明类似于文献[15]中的定理 7,因证明式(13)中 R_k^{UB} 单调性和收敛性的引理较多,篇幅较大,在此省略了证明。

因为 $R_k^{UB} \leqslant D_k$,用式(13)进行可调度性判定比式(11)更紧密。

3　FP 算法的可调度性判定

基于响应时间分析的 FP 算法可调度性判定的伪代码如下:

```
boolFpSchedulability(处理器个数 m,有限的任务集 τ){
    for each(任务集中所有任务 τκ,优先级自高向低排列){
        Rl = Cκ;
        // 迭代计算 τκ的响应时间上界
        while(true){
            Rl 代入式(13)右端的 Ri^UB,根据式(13),计算 τκ响应时间上界,赋值给 R2;
            if (R2 > Dκ)
                return 任务集不可调度;
            // b 是一个很小的值,判断是否还要迭代
            if (abs(Rl - R2) < b)
                break;//τκ可调度
                Rl = R2;}}
    return 任务集可调度;}
```

4　实验与结果分析

实验所用任务集由下列方法生成:设 u 为任务的平均利用率,U_i 遵循期望值 $\sigma_u = u$ 的指数分布;周期 T_i 在 $[10, 2\ 000]$ 上随机分布;最坏执行时间由 $C_i = U_i \times T_i$ 计算;截止期 D_i 在 $[C_i, T_i]$ 上随机分布。任务集共生成 100 000 个。

在生成了任务集后,测试本文描述的判定算法。为了比较实验结果,对文献[10]提供的算法进行同样测试。其实验结果如图 4 所示。图中表示了通过可调度性判定的任务集数量。

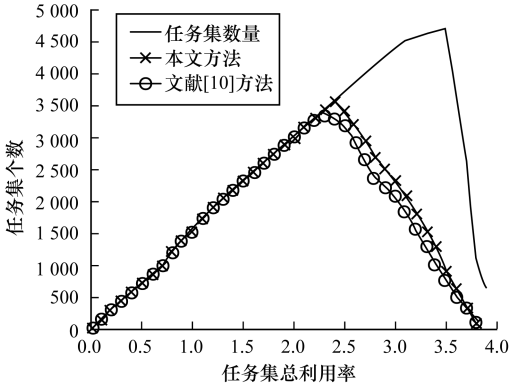

图 4 $\sigma_u = 0.3, m = 4$ 的实验数据

由图 4 可见,当利用率较低时,判定条件较为宽松,这一区间的任务集都能通过判定。但在任务利用率较高时,本文方法通过判定的任务集个数明显增加。而实际应用中常用的正是这一区间,实验说明了本文方法是一种更有效、紧密的方法。

5 结束语

本文针对文献[9]提出的 FP 算法判定方法,分析了实时任务带入作业数及其产生的干涉,着重分析了带入作业数对处理器需求的影响,分析中考虑了带入作业数对判定结果产生的影响,继而提出了一种较为严密的计算带入作业数及其干涉的方法。为了精确计算干涉,给出了基于截止期和响应时间的 2 种分析方法。在实际应用的多处理器实时系统中,任务数量一般都远多于处理器个数,使用本文方法能取得较好的效果。实验结果也表明了本文方法提高了通过判定的任务集数量。

参考文献

[1] DAVIS R, BURNS A. A Survey of Hard Real-time Scheduling for Multiprocessor Systems[J]. ACM Computing Surveys,2011,43(4):88-129.

[2] 彭 浩,韩江洪,陆 阳,等. 多处理器硬实时系统的抢占阈值调度研究[J]. 计算机研究与发展,2015,52(5):1177-1186.

[3] 李 杰,郭锐锋,邵志香,等. 面向多处理器的实时周期任务容错调度算法研究[J]. 小型微型计算机系统,2013,34(6):1253-1256.

[4] DAVIS R I,BURNS A. Priority Assignment for Global Fixed Priority Pre-emptive Scheduling in Multiprocessor Real-time systems[C]//Proceedings of Real-Time Systems Symposium. Washington D. C., USA: IEEE Press, 2009: 398-409.

[5] LEE J, SHIN I. Demand-based Schedulability Analysis for Real-time Multi-core Scheduling[J]. Journal of Systems & Software,2014,89(7):99-108.

[6] 孙景昊,孙景昶,关 楠,等. 偶发实时系统可调度性分析问题的整数规划方法[J]. 软件学报,2017,28(2):411-428.

[7] 张 晶,孙少杰,范洪博,等. 一种高回报的最小空闲时间优先实时调度改进算法[J]. 计算机工程,2017,43(3):57-61.

[8] 袁 野,晏 立. 基于负载计算的多处理器全局 EDF 判定方法[J]. 计算机工程,2012,38(12):287-290.

[9] BERTOGNA M. Real-time Scheduling Analysis for Multiprocessor Platforms[D]. Pisa, Italy: University of Sant'Anna,2007.

[10] BERTOGNA M,CIRINEI M,LIPARI G. Schedulability Analysis of Global Scheduling Algorithms on Multiprocessor Platforms[J]. IEEE Transactions on Parallel and Distributed Systems,2008,20(4):553-566.

[11] SUN Youcheng, LIPARI G. Response Time Analysis with Limited Carry-in for Global Earliest Deadline First Scheduling[C]//Proceedings of Real-Time Systems Symposium. Washington D. C., USA: IEEE Press, 2015: 130-140.

[12] LEE J,SHIN I. EDZL Schedulability Analysis in Real-time Scheduling[J]. IEEE Transactions on Software Engineering,2013,38(7):910-916.

[13] BERTOGNA M, BARUAH S. Test for Global EDF Schedulability Analysis[J]. Journal of Systems Architecture,2011,57(5):487-497.

[14] BARUAH S. Techniques for Multiprocessor Global Schedulability Analysis[C]//Proceedings of IEEE Real-Time Systems Symposium. Washington D. C., USA: IEEE Press,2007:119-128.

[15] BERTOGNA M, CIRINEI M. Response Time Analysis for Global Scheduled Symmetric Multiprocessor Platforms[C]//Proceedings of Real-Time Systems Symposium. Washington D. C., USA: IEEE Press,2007:149-158.

编辑 陆燕菲 顾逸斐