设备级 OS 中任务调度的 IO 抖动优化策略

刘 铮[1],赵 海[1],张 骞[2]
(1.东北大学信息科学与工程学院,辽宁沈阳 110819;2.东软集团技术战略与发展部,辽宁沈阳 110179)

摘 要: 为了减小任务调度产生的抖动对设备级操作系统的稳定性和可靠性的影响,提出了一种带有抢占阈值的任务分割模型 RJPS.根据抢占阈值对任务调度的抖动与延迟的影响及该模型下任务可调度性的分析,分别设计了基于固定优先级调度和动态优先级调度策略的最小阈值分配算法.仿真结果表明,该模型可以在保证任务集可调度的前提下明显减少任务的 IO 抖动.

关键词: 阈值分割模型; 抖动控制; 调度延迟; 嵌入式实时系统
中图分类号: TP316.2 **文献标识码:** A **文章编号:** 0372-2112 (2010) 11-2555-06

Optimization of IO Jitter in the Device-Level OS Scheduling

LIU Zheng[1], ZHAO Hai[1], ZHANG Qian[2]
(1. *College of Information Science and Engineering, Northeastern University, Shenyang, Liaoning* 110819, *China*;
2. *TSD of NeuSoft, Shenyang, Liaoning* 110179, *China*)

Abstract: In order to reduce the influence of the jitter of hard real-time task caused by scheduling to the stability and reliability of device-level operation system, a model with preemption segment threshold -RJPS is presented. According to the effect on jitter and latency brought by threshold and the analysis of the scheduling on this model, minimum threshold assigned algorithms are designed, which are applied for fixed-priority scheduling and fixed-priority scheduling separately. It is shown by simulation experiment that the threshold segmentation scheduling model can effectively reduce the IO jitter of tasks caused by scheduler in the typical scheduling algorithm.

Key words: threshold segmentation model; jitter control; scheduling latency; embedded real-time system

1 引言

设备级操作系统是由于工程实践的需要而诞生的,它属于硬实时操作系统,一般具有较小的体积.在大多数具有实时性需求的嵌入式系统中,系统的设计和实现以及支持它的设备和操作系统体系结构必须满足一定的限制条件来满足应用的定时约束,这对设备级操作系统提出了更高的要求.对于一些由硬实时任务控制的外部对象,外部设备如车床等大型机械的物理特性决定着开启、采样与运转过程中抖动不可避免[1,2],不能与操作系统的调度指令同步.映射到操作系统内部,则表现为硬实时任务不能在规定时间片上完成相关工作,任务存在输入抖动和 IO 抖动.实时操作系统中 IO 抖动具有叠加效果[3],随着时间的积累,抖动的存在使任务的可调度性不断降低,系统实时性和系统的控制性降低,甚至会导致失控现象和系统的崩溃[4,5].

延迟和抖动的随机性使抖动控制方法成为解决这一问题的主要手段.本文提出了一种基于抢占阈值的任务分割模型,通过阈值的合理分配,实现在保证任务集可调度的前提下对任务 IO 抖动的控制.

2 相关工作

实时控制应用中的抖动问题在过去的十几年里受到越来越多的关注,先后出现了很多用于降低任务抖动的理论.如 Nilsson 等人[6]和 Marti 等人[7]使用控制器补偿技术来减少延迟、抖动对系统性能产生的影响,延迟和抖动的控制将有利于补偿技术的实施;Cervin 等人[8]给出了更加逼近的 EDF 调度的最好情况响应时间下限,并对固定优先级调度及 EDF 调度的延迟和抖动进行了分析,但没有给出减小延迟和抖动的方法;Crespo[9]等人提出通过将任务分解为数据输入、处理、控制输出三个子任务来减小抖动,但以牺牲 IO 延迟为代价;Hoang 等人[10]和 Balbastre 等人[11]分别提出了两种在保证任务集可调度条件下最小化任务相对时限的方法来

收稿日期:2008-09-10;修回日期:2010-07-19
基金项目:国家自然科学基金(No.60973022)

达到减小延迟和抖动的目的,但该方法仅仅针对任务集中有少数抖动敏感任务的情况;Buttazzo 等人在文献[12]中的对比分析实验结果表明,与最小化相对时限的方法相比,非抢占调度策略对控制性能的重要指标——IO 延迟和抖动有更好的优化效果,但是存在可调度利用率过低的缺陷.

3 模型描述

在控制领域,针对传统的实时系统调度,现有的减少系统抖动的模型有以下几种:任务分割模型(RJTS, Reducing Jitter by Task Splitting)、修改截止时限模型(RJAD, Reducing Jitter by Advancing Deadlines)和修改可抢占性模型(RJNP, Reducing Jitter by Non-Preemption)[12].任务分割模型(RJTS)是指将任务的每个作业分割为 3 部分,作业的输入部分、作业的执行部分和作业的输出部分.其中作业的输入部分和输出部分不可抢占,作业的执行部分可以抢占.当使用这种策略时,作业的输入和输出部分常常在一个周期内被分开,这使得 IO 延迟增加,很可能使任务错过截止时限.修改可抢占性模型(RJNP)是指修改作业的可抢占性,将作业设置为不可抢占.这种模型可以减少作业的延迟和抖动,但是以牺牲系统的可调度性和系统的输入抖动为代价.修改截止时限模型(RJAD)是指在保证任务的可执行前提下,提前任务的截止时限,以达到减小作业的延迟和抖动的目的.由于静态优先级调度算法一般利用周期计算其优先级,因此这种方法一般用于优化动态优先级调度算法的延迟和抖动.但该模型修改了任务的截止时限,因此也就改变了任务的属性.

基于抢占阈值的任务分割模型(RJPS, Reducing Jitter by Preemption Segment threshold)是在不修改任务属性的前提下将任务的作业设置为部分可抢占.该模型描述如下:设实时系统的工作负荷模型为周期性任务集 τ,其中包括 n 个周期性任务 $\tau_1,\cdots,\tau_n$.每个周期性任务由一个五元组来表示,即 $\tau_i=(T_i,C_i,C_i^b,D_i,PS_i)$.其中 T_i 是任务 τ_i 的周期,D_i 是任务 τ_i 的相对时限,因为周期性任务的执行时间往往具有不确定性,为了更好的分析任务的抖动特性,所以引入了任务 τ_i 的最坏执行时间 C_i 和最好执行时间 C_i^b,且 $0<C_i^b\leqslant C_i\leqslant T_i$. PS_i 是任务 τ_i 的可抢占时间阈值,限定了任务 τ_i 可抢占的时间长度,且 $0\leqslant PS_i\leqslant C_i$.每个任务 τ_i 由可抢占部分开始,不可抢占部分结束.当任务的执行时间为 C_i 时,可抢占部分的长度为 PS_i,不可抢占部分的长度为 C_i-PS_i;当任务的执行时间为 C_i^b 时,可抢占部分的长度为 $\min(PS_i,C_i^b)$,不可抢占部分为 $C_i^b-\min(PS_i,C_i^b)$.

4 阈值对延迟与抖动的影响分析

4.1 延迟与抖动的特征描述

周期性任务的延迟与抖动特性参数定义如下:

$IOL_{i,k}$是作业 $J_{i,k}$的 IO 延迟,即作业 $J_{i,k}$实际开始执行到其执行结束的时间差,即 $IOL_{i,k}=f_{i,k}-s_{i,k}$.

IOJ_i 是周期性任务 τ_i 的 IO 抖动,是周期性任务 τ_i 的所有作业中最大 IO 延迟与最小 IO 延迟的差,即 $IOJ_i=\max_k(IOL_{i,k})-\min_k(IOL_{i,k})$.

$IOL_i^{\max}$ 是周期性任务的最大可能 IO 延迟,是最坏情况下,任务 τ_i 中所有作业 IO 延迟的最大值.

IOL_i^{min} 是周期性任务的最小可能 IO 延迟,是在最好情况下,任务 τ_i 中所有作业 IO 延迟的最小值.

$IOJ_i^{\max}$ 是周期性任务的最大可能 IO 抖动,是最大可能 IO 延迟与最小可能 IO 延迟的差,即 $IOJ_i^{\max}=IOL_i^{\max}-IOL_i^{\min}$.

4.2 固定优先级调度中的 IO 抖动分析

根据任务延迟与抖动的定义,任务的最大延迟和抖动取决于任务中所有作业的最大延迟和最小延迟,因此,我们只需对任务中作业的最大延迟和最小延迟进行分析.在基于阈值的末端非抢占模型中,阈值 PS_i 将任务 τ_i 分为可抢占部分和不可抢占部分,因此任务 τ_i 的最大可能 IO 延迟需要对这两部分分别计算.

根据时间需求分析[13],任务 τ_i 可抢占部分的最大可能 IO 延迟为

$$IOL_i^{pree}=PS_i+\sum_{k\in hp(i)}\left\lceil\frac{IOL_i^{pree}}{T_k}\right\rceil\cdot C_k,\quad i=1,2,\cdots,n \tag{1}$$

IOL_i^{pree} 是满足式(1)的最小解,可由式(1)以初值 $IOL_i^{pree(0)}=PS_i$ 迭代求解得到.

任务 τ_i 不可抢占部分的最大可能 IO 延迟为 $IOL_i^{nonp}=C_i-PS_i,i=1,2,\cdots,n$,因此,基于固定优先级调度的任务 τ_i 的最大可能 IO 延迟为

$$IOL_i^{\max}=IOL_i^{pree}+IOL_i^{nonp}=C_i+\sum_{k\in hp(i)}\left\lceil\frac{IOL_i^{pree}}{T_k}\right\rceil\cdot C_k \tag{2}$$

固定优先级调度的最小可能 IO 延迟应等于其最小可能响应时间,对于最小可能响应时间,文献[14]进行了分析,根据分析的结果,任务 $\tau_i(1\leqslant i\leqslant n)$可抢占部分的最小可能 IO 延迟为

$$IOL_i^{(b)pree}=\min(C_i^b,PS_i)+\sum_{k\in hp(i)}\left\lceil\frac{IOL_i^{(b)pree}-T_k}{T_k}\right\rceil\cdot C_k^b,\quad i=1,2,\cdots,n \tag{3}$$

$IOL_i^{(b)pree}$ 是以初值 $IOL_i^{(b)pree(0)}=T_i$ 开始迭代求得

的满足式(3)的最大解.

任务 τ_i 不可抢占部分的最小可能 IO 延迟为 $IOL_i^{(b)nonp}=\max(C_i^b,PS_i)-PS_i,i=1,2,\cdots,n$,因此,基于固定优先级调度的任务 τ_i 的最小可能 IO 延迟为

$$IOL_i^{\min}=IOL_i^{(b)pree}+IOL_i^{(b)nonp},i=1,2,\cdots,n \tag{4}$$

基于固定优先级调度的末端非抢占任务 τ_i 的最大可能 IO 抖动为

$$IOJ_i^{\max}=IOL_i^{\max}-IOL_i^{\min},i=1,2,\cdots,n \tag{5}$$

当 $PS_i\geqslant C_i^b$ 时,$IOL_i^{(b)nonp}=0$,所以

$$IOL_i^{\min}=IOL_i^{(b)pree}=C_i^b+\sum_{k\in hp(i)}\left\lceil\frac{IOL_i^{(b)pree}-T_k}{T_k}\right\rceil\cdot C_k^b \tag{6}$$

可见,此时 $IOL_i^{\min}$ 和 PS_i 无关,而 $IOL_i^{\max}$ 是 PS_i 的阶梯递增函数,因此最大可能 IO 抖动 $IOJ_i^{\max}$ 是 PS_i 的阶梯递增函数.

当 $PS_i<C_i^b$ 时,由式(3)得

$$IOL_i^{(b)pree}=PS_i+\sum_{k\in hp(i)}\left\lceil\frac{IOL_i^{(b)pree}-T_k}{T_k}\right\rceil\cdot C_k^b \tag{7}$$

由此可见 $IOL_i^{(b)pree}$ 是 PS_i 的阶梯递增函数.此时最大可能 IO 抖动为

$$\begin{aligned}IOJ_i^{\max}&=IOL_i^{\max}-IOL_i^{\min}\\&=\sum_{k\in hp(i)}\left\lceil\frac{IOL_i^{pree}}{T_k}\right\rceil\cdot C_k-\sum_{k\in hp(i)}\left\lceil\frac{IOL_i^{(b)pree}}{T_k}-1\right\rceil\cdot C_k^b\\&\quad+(C_i-C_i^b)\end{aligned} \tag{8}$$

式(8)的左边第一项和第二项都是 PS_i 的阶梯递增函数,因此不能确定 $IOJ_i^{\max}$ 是 PS_i 的阶梯递增函数,但由于左边第一项的递增系数 C_k 大于左边第二项的递增系数 C_k^b,所以当 PS_i 减小时,最大可能 IO 抖动 $IOJ_i^{\max}$ 呈减小趋势.

4.3　动态优先级调度的 IO 延迟与抖动分析

首先,根据基于 EDF 调度的时间需求分析,任务 τ_i 可抢占部分最大可能 IO 延迟为

$$IOL_i^{pree}(a)=\max\{PS_i,L_i^{pree}(a)-a\},i=1,2,\cdots,n \tag{9}$$

其中

$$L_i^{pree}(a)=W_i(a,L_i^{pree}(a))+\left\lfloor\frac{a}{T_i}\right\rfloor\cdot C_i+PS_i+\max_{D_j>D_i}(C_j-PS_j) \tag{10}$$

忙周期长度 $L_i^{pree}(a)$ 是满足式(10)的最小解,可由式(10)以初值 $L_i^{pree(0)}=0$ 迭代求解得到,而其中的高优先级作业负荷 $W_i(a,t)$ 为式(11)所描述.

$$W_i(a,t)=\sum_{\substack{j\neq i\\D_j\leqslant a+D_i}}\min\left\{\left\lceil\frac{t}{T_j}\right\rceil,1+\left\lfloor\frac{a+D_i-D_j}{T_j}\right\rfloor\right\}\cdot C_j \tag{11}$$

此外,任务 τ_i 不可抢占部分的最大可能 IO 延迟为 $IOL_i^{nonp}=C_i-PS_i,i=1,2,\cdots,n$,因此,基于 EDF 调度的任务 τ_i 的最大可能 IO 延迟为

$$\begin{aligned}IOL_i^{\max}&=IOL_i^{pree}(a)+IOL_i^{nonp}\\&=\max\{PS_i,L_i^{pree}(a)-a\}+(C_i-PS_i)\end{aligned} \tag{12}$$

文献[2]还对 EDF 调度的最小可能响应时间进行了分析,而 EDF 调度的最小可能 IO 延迟应等于其最小可能响应时间,因此任务 $\tau_i(1\leqslant i\leqslant n)$ 可抢占部分的最小可能 IO 延迟为

$$IOL_i^{(b)pree}=\min(C_i^b,PS_i)+\sum_{\forall j:D_j<IOL_i^{(b)pree}}\left\lceil\frac{\min\{IOL_i^{(b)pree},D_i-D_j\}}{T_j}-1\right\rceil\cdot C_j^b \tag{13}$$

$IOL_i^{(b)pree}$ 是以初值 $IOL_i^{(b)pree(0)}=T_i$ 开始迭代求得的满足式(13)的最大解.

任务 τ_i 不可抢占部分的最小可能 IO 延迟为 $IOL_i^{(b)nonp}=\max(0,C_i^b-PS_i)$,因此,基于 EDF 调度的任务 τ_i 最小可能 IO 延迟为

$$\begin{aligned}IOL_i^{\min}&=IOL_i^{(b)pree}+IOL_i^{(b)nonp}\\&=C_i^b+\sum_{\forall j:D_j<IOL_i^{(b)pree}}\left\lceil\frac{\min(IOL_i^{(b)pree},D_i-D_j)}{T_j}-1\right\rceil\cdot C_j^b\end{aligned} \tag{14}$$

下面对基于 EDF 调度的任务 τ_i 的最大可能 IO 抖动 $IOJ_i^{\max}=IOL_i^{\max}-IOL_i^{\min}$ 进行分析:

(1)若 $PS_i\geqslant L_i^{pree}(a)-a$ 并且 $C_i^b\leqslant PS_i$,则式(13)与 PS_i 无关,因此 $IOJ_i^{\max}$ 与 PS_i 无关;(2)若 $PS_i\geqslant L_i^{pree}(a)-a$ 并且 $C_i^b>PS_i$,则 $IOJ_i^{\max}=C_i-IOL_i^{\min}$,根据式(13)和(14)可知 $IOL_i^{\min}$ 是 PS_i 的阶梯递增函数,因此 $IOJ_i^{\max}$ 是 PS_i 的阶梯递增函数;(3)若 $PS_i<L_i^{pree}(a)-a$ 并且 $C_i^b\leqslant PS_i$,则式(13)与 PS_i 无关,即 $IOL_i^{\min}$ 与 PS_i 无关.由前面分析可知,$IOL_i^{\max}$ 是 PS_i 的阶梯递增函数,因此 $IOJ_i^{\max}$ 是 PS_i 的阶梯递增函数;(4)若 $PS_i<L_i^{pree}(a)-a$ 并且 $C_i^b>PS_i$,

$$\begin{aligned}IOJ_i^{\max}&=IOL_i^{\max}-IOL_i^{\min}\\&=\sum_{\substack{j\neq i\\D_j\leqslant a+D_i}}\min\left\{\left\lceil\frac{L_i^{pree}(a)}{T_j}\right\rceil,1+\left\lfloor\frac{a+D_i-D_j}{T_j}\right\rfloor\right\}\cdot C_j\\&\quad+\left(1+\left\lfloor\frac{a}{T_i}\right\rfloor\right)\cdot C_i+b-a-C_i^b\\&\quad-\sum_{\forall j:D_j<IOL_i^{(b)pree}}\left\lceil\frac{\min(IOL_i^{(b)pree},D_i-D_j)}{T_j}-1\right\rceil\cdot C_j^b\end{aligned} \tag{15}$$

公式(15)的第一项和最后一项都是 PS_i 的阶梯递增函数,但由于第一项的递增系数 C_j 大于最后一项的

递增系数 $C_j^{(b)}$,第一项的忙周期长度 $L_i^{pree}(a)$大于最后一项的可抢占部分响应时间 $IOL_i^{(b)pree}$ 长度,所以当 PS_i 减小时,最大可能 IO 抖动 $IOJ_i^{\max}$ 呈减小趋势.

5 阈值分配算法

由上面的分析可知,带有 PS 阈值的末端非抢占模型可以在一定程度上减小任务的 IO 抖动,而且阈值越小,优化的程度越明显.但是,对于实时系统来说,保证其任务的可调度性是衡量其性能的首要指标.因此,在确定阈值的过程中,不仅要考虑到对任务抖动的优化,还要保证系统的可调度性.根据 RM 算法的充要条件和 EDF 算法的充分条件,推导出能够保证系统可调度的最小可抢占阈值分配算法.

5.1 固定优先级调度系统中的阈值分配算法

在固定优先级调度中,设任务按优先级降序进行索引,即索引 $i(1\leqslant i\leqslant n)$越大任务 τ_i 的优先级越低.因为任务的开始执行可能受到低优先级任务不可抢占部分的阻塞,所以带有阈值约束的任务 τ_i 的可抢占部分的时间需求函数为

$$w_i^{pree}(t,PS_1,PS_2,\cdots,PS_n)=\max_{j=i+1}^{n}(C_j-PS_j)+PS_i+\sum_{k=1}^{i-1}\left\lceil\frac{t}{T_k}\right\rceil\cdot C_k \tag{16}$$

由于任务 τ_i 具有末端非抢占性,且非抢占部分的最大可能执行时间为 C_i-PS_i,因此保证任务 τ_i 可调度的充要条件为:存在 $t\in S_i$,使 $w_i^{pree}(t-(C_i-PS_i),PS_1,PS_2,\cdots,PS_n)+C_i-PS_i\leqslant t$ 成立.其中 S_i 为检测点集合,且

$$S_i=\left\{t\,\middle|\,t=u\cdot p_v,v=1,2,\cdots,i,u=1,2,\cdots,\left\lfloor\frac{D_i-(C_i-PS_i)}{T_v}\right\rfloor\right\} \tag{17}$$

即,存在 $t\in S_i$,使

$$\max_{j=i+1}^{n}(C_j-PS_j)\leqslant t-C_i-\sum_{k=1}^{i-1}\left\lceil\frac{t-(C_i-PS_i)}{T_k}\right\rceil\cdot C_k \tag{18}$$

成立.

由式(18)可知,任务 τ_i 的可调度性与高优先级任务的可抢占时间阈值无关,因此阈值可从高优先级到低优先级依次分配.因为最高优先级任务 τ_1 不会被其它任务抢占,为了方便计算,可分配其阈值 $PS_1^{\min *}=C_1$.当 PS_i 分配后,即 $PS_i=PS_i^{\min *}$,则式(18)中的 PS_j 在等号成立时取得最小值,即保证任务 τ_i 可调度的 PS_{i+1}的最小值为

$$PS_{i+1}^{\min}=C_{i+1}-\max_{t\in S_i}\left(t-C_i-\sum_{k=1}^{i-1}\left\lceil\frac{t-(C_i-PS_i^{\min^*})}{T_k}\right\rceil\cdot C_k\right) \tag{19}$$

因此保证所有高优先级任务 $\tau_j(j=1,2,\cdots,i)$可调度的 PS_{i+1}的最小值为

$$PS_{i+1}^{\min *}=\max_{j=1}^{i}(PS_{j+1}^{\min}) \tag{20}$$

根据式(20),可对 $\tau_i(i=1,2,\cdots,n-1)$依次分配所有可抢占时间阈值.算法的具体描述如下:

(1)输入 $n,p_i,C_i,D_i,C_i^b(i=1,2,\cdots,n)$;(2)令 $PS_1^{\min *}=C_1$,分配任务 τ_1 的可抢占时间阈值 $PS_1=PS_1^{\min *}$;(3)初始化 $i=1$;(4)根据式(17)计算检测点集合 S_i;(5)根据式(19)和(20)计算 $PS_{i+1}^{\min^*}$,分配任务 τ_{i+1} 的可抢占时间阈值 $PS_{i+1}=PS_{i+1}^{\min^*}$;(6)$i=i+1$,返回步骤 4 继续分配,直到所有阈值分配完毕;(7)输出可抢占时间阈值的分配结果 $PS_1,PS_2,\cdots,PS_n$;(8)算法结束.

该算法给出了每个任务可抢占时间阈值的一种最优分配,并量化出该阈值分配下任务的最大可能 IO 延迟和最大可能 IO 抖动.因为检测点集合 S_i 中元素的最多个数为 $\left\lfloor\frac{T_n}{T_1}\right\rfloor\cdot n$,所以对每个任务计算式(19)的时间复杂度为 $O(\left\lfloor\frac{T_n}{T_1}\right\rfloor\cdot n)$,因此该可抢占时间阈值分配算法的时间复杂度为 $O(\left\lfloor\frac{T_n}{T_1}\right\rfloor\cdot n^2)$.

5.2 动态优先级调度系统中的阈值分配算法

在动态优先级调度算法 EDF 中,设任务按相对时限升序进行索引,即索引 $i(1\leqslant i\leqslant n)$越大任务 τ_i 的相对时限越大.因为在 EDF 调度系统中只有相对时限大的任务能够阻塞相对时限小的任务,所以带有可抢占时间阈值约束的任务 τ_i 的可抢占部分的时间需求函数为:

$$w_i^{pree}(a,PS_1,PS_2,\cdots,PS_n)=W_i(a,L_i^{pree}(a))+\left\lfloor\frac{a}{T_i}\right\rfloor\cdot C_i+PS_i+b_i,\quad b_i=\max_{D_j>D_i}(PS_j) \tag{21}$$

由于任务 τ_i 的末端非抢占性,且非抢占部分的最大可能执行时间为 C_i-PS_i,因此保证任务 τ_i 可调度的充要条件为:存在 $a\in S$,使

$$w_i^{pree}(a,PS_1,PS_2,\cdots,PS_n)+C_i-PS_i\leqslant a+D_i \tag{22}$$

成立,即存在 $a\in S$ 使

$$\max_{j=i+1}^{n}(C_j-PS_j)\leqslant a+D_i-W_i(a,L_i^{pree}(a))+\left\lfloor\frac{a}{T_i}\right\rfloor\cdot C_i+PS_i+b_i,$$

$$b_i = \max_{D_j > D_i}(C_j - PS_j) \tag{23}$$

成立.

从公式(23)中,可以看出在计算 PS_i 的过程中,需要知道 PS_j 的值.因此,从充要条件无法推导出 PS_i 的值.PS_i 的分配公式可以从 EDF 可调度的充分条件推出.EDF 的充分条件为:

$$\sum_{k=1}^{n}\frac{C_k}{\min(D_k,T_k)}+\frac{b_i}{\min(D_i,T_i)}\leqslant 1, b_i=\max_{D_j>D_i}(C_j-PS_j) \tag{24}$$

则 b_i 的计算公式为

$$b_i \leqslant \left(1-\sum_{k=1}^{n}\frac{C_k}{\min(D_k,T_k)}\right)\cdot\min(D_i,T_i) \tag{25}$$

根据上面的公式,首先确定每个任务被阻塞的时间 b_i,然后根据 b_i 的值计算相对时限比任务 τ_i 大的任务 τ_j 的阈值 PS_j.依照此方法对任务 $\tau_j(j=2,3,\cdots,n)$的阈值进行分配.算法可描述如下:

(1)输入 $n, T_i, C_i, D_i, C_i^b(i=1,2,\cdots,n)$;(2)令 $PS_1^{\min^*}=C_1$,分配任务 τ_1 的可抢占时间阈值 $PS_1=PS_1^{\min^*}$;(3)初始化 $i=1$;(4)根据式(25)计算 b_i^*;(5)将 $b_{i+1}^*,\cdots,b_n^*$ 赋值为 b_i^*;(6)令 $i=i+1$,返回步骤(4)继续分配,直到所有 b_i^* 计算完毕;(7)根据 $PS_i=C_i-b_i^*$,依次计算 $\tau_i(i=1,2,\cdots,n)$的可抢占时间阈值 PS_i;(8)输出可抢占时间阈值的分配结果 $PS_1,PS_2,\cdots,PS_n$;(9)算法结束.

因为式(25)中 b_i 的最大值由 EDF 可调度的充分条件求得,因此该算法给出了每个任务可抢占时间阈值的一种次优分配,由式(24)和(25)可知,该分配算法的时间复杂度为 $O(n)$.

6　仿真实验与优化效果分析

为了验证本文所提出的模型和算法对优化实时系统中任务抖动的有效性,首先在仿真平台 TORSCHE Scheduling Toolbox for Matlab 中编程实现本文提出的带有可抢占时间阈值的任务模型及支持该任务模型的调度机制,然后按如下方法生成负载任务集:每个任务的周期 T_i 由相同分布的随机变量在 10 到 100 个时间单元之间产生,每个任务的最坏执行时间 C_i 在 1 到 T_i 个时间单元之间随机产生后,乘以一个共同的因子 $U/\sum_{i=1}^{n}\frac{C_i}{T_i}$ 来控制任务的总利用率 U 为指定数值.设定实时任务数 $n=7$,并分别对 $U=0.1,0.2,\cdots,0.9$ 的情况随机产生 500 组任务集,仿真时间设定为 1000 个时间单元.

6.1　RM 调度的抖动优化效果及分析

在上述实验平台上,对基于 RM 调度的 RJTS、RJNP 和本文提出的 RJPS 策略分别进行仿真,并求得各调度策略下的平均 IO 延迟和 IO 抖动.如图 1、图 2 所示.

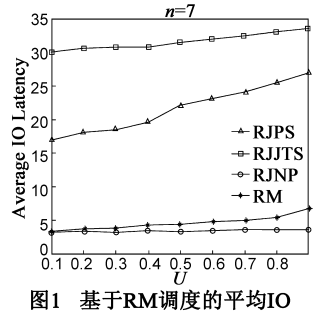


图1　基于RM调度的平均IO延迟对比

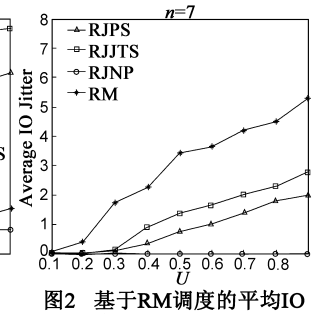


图2　基于RM调度的平均IO抖动对比

根据实验数据,从定量的角度对各策略优化效果作进一步分析.以原始的 RM 算法作为基本参照,各策略的优化效果见表 1 所示.

表 1　各优化模型对 RM 调度的优化效果

	RJNP	RJTS	RJPS
IO 抖动	降低至 0	降低 53.4%	降低 78.1%
IO 延迟	减小 10.9%	增加 5.4 倍	增加 2.9 倍
不可调度性	接近 20%	低于 5%	低于 5%

与其他两种策略相比,RJPS 策略在最大可能保证系统可调动的前提下,可以较好的优化 IO 抖动.但是,随着系统负载的增加,优化空间减小,优化效果受到一定限制.这种策略所付出的代价是任务的平均 IO 延迟(时间单元)略有所增加.

6.2　EDF 调度的抖动优化效果及分析

接下来,在动态优先级 EDF 调度算法的基础上分别对 RJTS、RJAD 和 RJNP 三种调度策略进行了仿真实验.实验结果如图 3、图 4 所示.

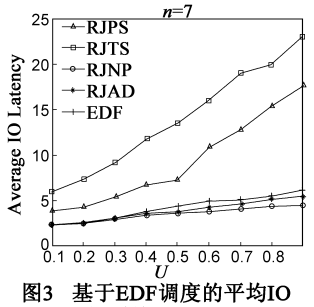


图3　基于EDF调度的平均IO延迟对比

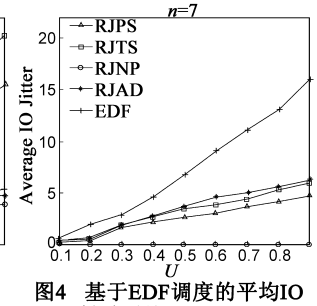


图4　基于EDF调度的平均IO抖动对比

以经典的 EDF 调度算法为参考,根据实验得到的数据对这四种策略进行定量分析,各策略的具体优化效果见表 2.

表 2　各优化策略对 EDF 算法的优化效果

	RJNP	RJTS	RJAD	RJPS
IO 抖动	减小至 0	减小 71.8%	减小 67.3%	减小 77.5%
IO 延迟	减小 3.9%	增加 3.4 倍	减小 1.1%	增加 1.3 倍
不可调度性	接近 17%	低于 5%	低于 5%	低于 5%

可见,与其他三种策略相比,本文提出的 RJPS 抖动

优化策略可以较好的优化 IO 抖动,并且 IO 延迟不会有较大的增加,系统的可调度性也能尽量得到保证,可以说 RJPS 是减少 IO 抖动的最安全策略.

7 结论

在复杂的嵌入式实时控制系统中,调度产生的任务抖动对控制系统的性能具有较大的负面影响,为了在保证任务集可调度的前提下,控制硬实时任务的抖动,改善嵌入式系统对外界环境的控制性能,本文提出一个带有可抢占时间阈值的末端非抢占任务分割模型,并分别对固定优先级算法和 EDF 算法调度的系统,分析了硬实时任务的 IO 抖动特性,得出了最小化可抢占时间阈值的参数优化目标.

对基于固定优先级算法调度的系统,本文给出了一种可调度性约束条件下时间复杂度为 $O(\left\lfloor \frac{T_n}{T_1} \right\rfloor \cdot n^2)$的最优阈值分配算法;而对于 EDF 调度,因其临界时刻检测点所对应的时限忙周期长度受到可抢占时间阈值的影响,因此通过 EDF 可调度的充要条件分配最优可抢占时间阈值的问题是一个 NP 难题,因此本章使用 EDF 可调度的充分条件给出一种时间复杂度为 $O(n)$的次优分配.

仿真实验表明,和现有其它策略相比,本文提出的 IO 抖动优化策略可以在最大保证系统可调度的前提下有效的降低硬实时任务 IO 抖动.但作为代价,任务的 IO 延迟略有增加.因此该模型可以应用在对 IO 延迟要求不高但对 IO 抖动要求很高的实际应用中.

参考文献:

[1] Grenier M N N. New on-line preemptive scheduling policies for improving real-time behavior [A]. Proceedings of the 10th IEEE International Conference on Emerging Technologies and Factory Automation [C]. Catania, Italy: IEEE Press, 2005. 322 – 329.

[2] Cervin A, Lincoln B, Eker J E A. The jitter margin and its application in the design of real-time control systems [A]. Proceedings of the 10th International Conference on Real-Time and Embedded Computing Systems and Applications [C]. Gothenburg, Sweden, 2004. 1 – 9.

[3] Balakrishnan A. On the problem of time jitter in sampling [J]. IEEE Transactions on Information Theory, 1962, 8(3): 226 – 236.

[4] P M. Analysis and Design of Real-Time Control Systems with Varying Control Timing Constraints [D]. Barcelona, Spain: Technical University of Catalonia, 2002.

[5] A C. Integrated Control and Real-Time Scheduling [D]. Lund, Sweden, 2003.

[6] Nilsson J, Bernhardsson B W B. Stochastic analysis and control of real-time systems with random time delays [J]. Automatica (Journal of IFAC), 1998, 34(1): 57 – 64.

[7] Marti P, Fohler G, Ramamritham K E A. Jitter compensation for real-time control systems [A]. Proceedings of the 22rd IEEE Real-Time System Symposium [C]. London, UK, 2001. 39 – 48.

[8] Cervin A, Lincoln B, Eker J E A. The jitter margin and its application in the design of real-time control systems [A]. Proceedings of the 10th International Conference on Real-Time and Embedded Computing Systems and Applications [C]. Gothenburg, Sweden, 2004. 1 – 9.

[9] Crespo A, Ripoll I A P. Reducing delays in RT control: the control action interval [A]. Proceedings of the 14th IFAC World Congress[C]. Beijing, 1999. 1 – 6.

[10] Hoang H, Buttazzo G, Jonsson M E A. Computing the minimum EDF feasible deadline in periodic systems [A]. Proceedings of the 12th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications [C]. Sydney, Australia, 2006. 125 – 134.

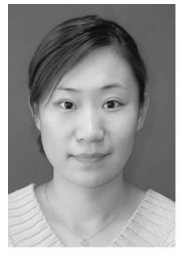
[11] Balbastre P, Ripoll I C A. Optimal deadline assignment for periodic real-time tasks in dynamic priority systems [A]. Proceedings of the 18th Euromicro Conference on Real-Time Systems (ECRTS 2004) [C]. Dresden, Germany, 2006. 65 – 74.

[12] Buttazzo G C A. Comparative assessment and evaluation of jitter control methods [A]. Proceedings of the 15th International Conference on Real-Time and Network Systems (RTNS2007) [C]. Nancy, France, 2007. 1 – 10.

[13] Jane W S Liu. Real-Time Systems [M]. Arrangement with the original publisher, Pearson Education, 2000. 113 – 115.

[14] Redell O M T. Exact best-case response time analysis of fixed priority scheduled tasks [A]. The IEEE 14th Euromicro Conference on Real-Time Systems (ECRTS'02) [C]. Vienna, Austria: IEEE Computer Society, 2002. 165 – 172.

作者简介:



刘 铮 女,1979 年 11 月出生于辽宁沈阳,东北大学讲师.分别于 2002、2005、2010 年在东北大学获得学士、硕士、博士学位,主要研究方向为嵌入式实时系统、复杂网络等.
E-mail: liuzheng@mail.neu.edu.cn

赵 海 男,东北大学教授,博士生导师.1959 年出生于辽宁沈阳.主要研究方向为嵌入式系统、复杂网络、传感器网络等.

张 骞 男,资深软件工程师.1979 年 1 月出生于山东金乡,2001 年在青岛大学获学士学位,2003 年和 2007 年在东北大学分别获得硕士学位和博士学位.主要研究方向为嵌入式系统、对等计算.