定时器驱动的 RM 调度机制建模及其性能优化

王济勇 赵 海 林 涛 王小英 王金东 韩光洁

(东北大学信息科学与工程学院计算机系统研究所 沈阳 110004)

摘 要 在 Katcher 等人对定时器驱动的 RM(Rate Monotonic)调度机制研究的基础上,通过对该机制下实时任务抢占行为的分析,建立了周期性任务的抢占模型,给出了直接抢占发生的充分必要条件,据此确定了任务间的抢占关系,进而精确了可调性的判定条件,然后讨论了系统的平均响应时间.依据此抢占模型,受生物界寄生现象的启发,提出了一个改善嵌入式系统实时性能的方法,将获取机制和利用机制分离,屏蔽了复杂优化计算对目标嵌入式系统性能的负面影响.最后,通过实验验证了该方法在改善抢占关系、减少抢占开销和增强系统可调度性方面的有效性,结果表明可调度利用率可以提高 0.25%～6.64%.

关键词 RM 调度算法;嵌入式系统;实时性能;定时器驱动;进化策略

中图法分类号 TP302

Modeling Timer-Driven RM Scheduling Mechanism to Improve Its Real-Time Performance

WANG Ji-Yong ZHAO Hai LIN Tao WANG Xiao-Ying WANG Jin-Dong HAN Guang-Jie

(*Institute of Computer System, School of Information Science & Engineering, Northeastern University, Shenyang* 110004)

Abstract Katcher *et al*. discussed the timer-driven RM (Rate Monotonic) scheduling mechanism and built its model. This model only shows the sufficient condition of schedulability. Underlying the model and analyzing preemptions of real-time tasks under this scheduling mechanism, this paper presents a preemption model of periodic tasks to improve the model presented by Katcher *et al*.. The preemption model gives the necessary and sufficient conditions of preemptions and schedulability, identifies the order of preemptions, derives the equations for computing overheads incurred by preemptions and average response time. Based on the preemption model and elicited by parasitism of biology, this paper shows a method of optimizing real-time performance of the embedded systems under the scheduling mechanism. The method separates the utilizing mechanism and the producing mechanism, shields the side-effects of optimization computing to the real-time performance of the target systems. Then, the experiments are used to validate the model and method in improving the order of preemptions, reducing the preemption overheads, and making a non-schedulable task set schedulable. The results show schedulable utilization can be improved 0.25%～6.64%.

Keywords RM scheduling algorithm; embedded system; real-time performance; timer driven; evolutionary strategy

---

收稿日期:2003-07-28;修改稿收到日期:2004-11-22.**王济勇**,男,1975 年生,博士研究生,主要研究方向为嵌入式系统和嵌入式 Internet. E-mail:wang.jiyong@zte.com.cn. **赵 海**,男,1959 年生,博士,教授,博士生导师,主要研究方向为嵌入式 Internet 和信息融合. **林 涛**,男,1975 年生,博士研究生,主要研究方向为嵌入式 Internet. **王小英**,女,1975 年生,博士研究生,主要研究方向为嵌入式 Internet. **王金东**,男,1976 年生,博士研究生,主要研究方向为嵌入式 Internet. **韩光洁**,男,1972 年生,博士研究生,主要研究方向为嵌入式 Internet.

1 引 言

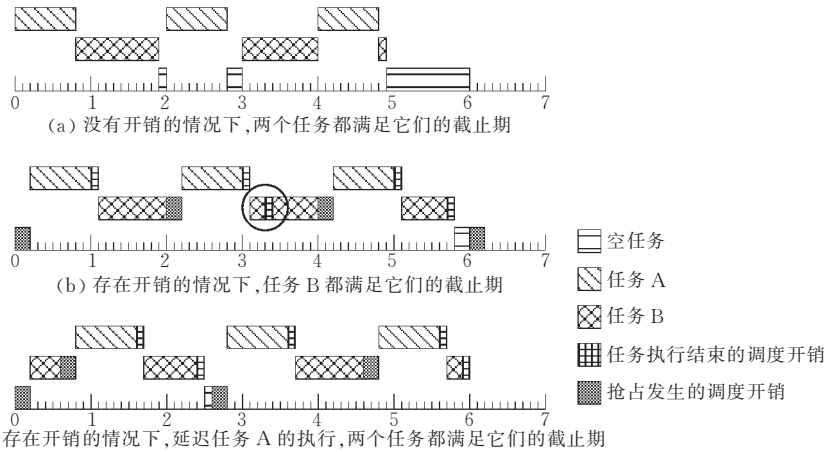
Liu 和 Layland[1] 所做的开创性工作提供了能够确定一个周期性任务集能否满足其中各个任务的硬时限约束的理论的分析方法.但是,这些理想化的理论基本上都忽略了在非理想资源上调度一个任务集所导致的实现开销.实际上,实时调度理论与其在构建系统上运行的操作系统内核存在着很大的差距,弥合这些差距是近年来实时嵌入式系统研究领域的热点问题[2~5].

对于根据任务的周期为每个任务分配唯一优先级的 RM 调度算法,Liu 和 Layland 在文献[1]中给出了判断可调度性的充分条件.在实际的工程实现中,RM 调度算法具有不同的实现机制,Katcher 等人对其进行了分类,考虑了实时嵌入式系统实现的开销和阻塞,建立了桥接理想理论模型与工程实际的模型[6].这其中涉及到两个重要概念,即开销和阻塞,Katcher 等人对它们的定义如下.

定义 1. 开销是内核代表特定任务完成服务所花费的时间,如唤醒或中断这个任务.

定义 2. 阻塞是当更高优先级任务被阻止运行而花费在内核或应用程序任务上的时间,即优先级倒置所持续的时间.

对于采用抢占式调度机制的嵌入式系统,实现的开销、阻塞和严格约束优先执行高优先级的任务导致资源利用率和平均性能的降低.下面是一个启发性例子的调度顺序图,如图 1.为了简单起见,仅考虑两个任务:A 和 B 组成的系统,它们的周期和最坏情况执行时间分别为 $(2,0.8)_A$ 和 $(3,1.1)_B$,相对截止期与其周期相等.假定存在开销时,任务执行结束的调度开销为 0.1,抢占发生的调度开销为 0.2.在没有任何调度开销的情况下,任务 A 和任务 B 的运行如图 1 中的(a);通常情况下,存在调度开销的环境中,任务 A 和任务 B 的运行如图 1 中的(b),被圆圈标出的部分表明任务 B 的这次执行没有满足其时限约束;存在调度开销的环境中,给高优先级的任务 A 附加了非零的相位,任务 A 和任务 B 的运行如图 1 中的(c).从图 1 中可以看出,适当地延迟高优先级任务的执行能够保证系统中的任务都能够满足它们的时限约束,提高系统的可调度能力,进而可以提高系统的实时性能.因此,如何延迟高优先级任务的执行和延迟多少是本文的研究内容.



(a) 没有开销的情况下,两个任务都满足它们的截止期

(b) 存在开销的情况下,任务 B 都满足它们的截止期

(c) 存在开销的情况下,延迟任务 A 的执行,两个任务都满足它们的截止期

图 1 一个启发性例子的调度顺序图

为了确定延迟高优先级任务执行的方法,结合其他人的研究成果[7,8],深入地分析了采用定时器驱动的 RM 调度机制的实时系统运行中抢占发生过程,建立了该调度机制下,周期性任务集的抢占模型 TDRMPModel,讨论了直接抢占发生的充分必要条件,确定了各实时任务每次执行的历经时间,从而给出了可调度性判断的充分必要条件.进而,得出了刻画系统的实时性能的两个指标——抢占导致的额外开销和平均响应时间的计算公式.依据此抢占模型,获得了可以延迟高优先级任务执行以改善嵌入式系统实时性能的方法.

嵌入式目标系统因其资源所限,通常工作在近乎满负荷的状态下,由其自身完成优化计算在实际中不可行.为了能够计算优化的延迟并利用这些优化值,在模型 TDRMPModel 的基础上,受生物界寄生现象的启发,我们提出了一个优化该调度机制下嵌入式系统实时性能的方法.分离了优化值利用机制和获取机制,作为获取机制的优化计算过程在桌面计算机上完成,而嵌入式目标系统仅保留能够利用这些优化计算结果的利用机制,从而屏蔽了复杂

优化计算对嵌入式目标系统实时性能的负面影响.以硬实时系统为例,本文将可调度性和抢占导致的额外开销为优化目标,随机选择一组实时任务集,实验结果表明,该方法使抢占导致的额外开销减少 9.68%～55.0%,可调度利用率提高 0.25%～6.64%.而这个方法的最大贡献在于,在不采用其它优化方法的情况下,能够改变一个轻微过载嵌入式系统的可调度性.

2　周期性任务的抢占模型

Katcher 等人建立的定时器驱动的 RM 调度机制模型[6]给出了判定该调度机制可调度性的充分条件.为了能够更精确地描述该机制的行为,在文献[6]模型的基础上,建立了周期性任务的抢占模型 TDRMPModel,明确了直接抢占行为发生的充分必要条件,并从数学上定量地刻画了抢占行为和中断事件与优先级、周期、执行时间、截止期、相位等实时任务属性的关系.在此基础上,这个模型给出了该调度机制下周期性任务集可调度性判断的充分必要条件.然后,得出了描述系统实时性能的两个指标(即抢占导致的额外开销和平均响应时间)的计算公式.为了描述方便起见,所需的符号及其定义如下:

一个实时任务集 T_{set} 由 n 个周期性任务组成,即 $\tau_0,\tau_1,\cdots,\tau_{n-1}$,按优先级降序排列.

每个任务 τ_i 都有一个周期 T_i,一个相对截止期 D_i,相对截止期与周期相等即 $T_i=D_i$,一个最坏执行时间 C_i,一个固定的优先级 P_i,一个相对启动时间 r_i 和一个相位 I_i,$i=0,1,\cdots,n-1$.

$e_{i,u}$ 是任务 τ_i 第 u 次执行的执行历经时间,指本次执行从启动到执行结束的时间间隔,包括所有在本次执行期间抢占它的任务的执行历经时间和系统的额外开销,它的数值在任务的执行过程是变化的,在时刻 t_0 它的计算值记为 $e_{i,u}|_{t0}$,$i=0,1,\cdots,n-1$,$u=0,1,\cdots,\lceil t/T_i\rceil-1$.为描述方便,给出以下定义:

T_{timer} 为系统中驱动调度的定时器周期;

t 为系统的运行时间,$t\geq 0$;

$w_{i,u}$ 是任务 τ_i 第 u 次执行被抢占的次数,$i=0,1,\cdots,n-1$,$u=0,1,\cdots,\lceil t/T_i\rceil-1$.

$q_{i,u}$ 是任务 τ_i 第 u 次执行被非抢占中断请求中断的次数,$i=0,1,\cdots,n-1$,$u=0,1,\cdots,\lceil t/T_i\rceil-1$.

$Pree_{i,u}$ 是任务 τ_i 第 u 次执行的抢占集,这是一个有序集,其中每个元素是一个序对$\langle j,v\rangle$,这个序对表示任务τ_i第 u 次执行时被任务 τ_j 第 v 次执行抢占,$Pree_{i,u}$ 中的元素顺序与抢占发生的顺序一致,$i=0,1,\cdots,n-1$,$j=0,1,\cdots,i-1$,$u=0,1,\cdots,\lceil t/T_i\rceil-1$,$v=0,1,\cdots,\lceil t/T_j\rceil-1$.

$NPIrq_{i,u}$ 是任务 τ_i 第 u 次执行的非抢占中断集,这是一个有序集,其中每个元素是一个序对$\langle j,v\rangle$,该序对表示任务 τ_i 第 u 次执行时被任务 τ_j 第 v 次执行请求非抢占中断,$NPIrq_{i,u}$ 中元素的顺序与中断发生的顺序一致,$i=0,1,\cdots,n-1$,$j=i+1,i+2,\cdots,n-1$,$u=0,1,\cdots,\lceil t/T_i\rceil-1$,$v=0,1,\cdots,\lceil t/T_j\rceil-1$.

$Irq_{i,u}$ 是任务 τ_i 第 u 次周期性请求发生的时刻,有的文献中将其称为启动时间,即 $Irq_{i,u}=I_i+u\cdot T_i$,$i=0,1,\cdots,n-1$,$u=0,1,\cdots,\lceil t/T_i\rceil-1$.

$r_{i,u}$ 是任务 τ_i 第 u 次执行的实际开始执行时刻,有的文献中称为开始时间.由于在时刻 $I_i+u\cdot T_i$ 可能存在更高优先级的任务在运行、系统处于临界区中不可中断或者触发调度的定时器中断未发生而阻塞,实际的开始时间就会被延迟,$i=0,1,\cdots,n-1$,$u=0,1,\cdots,\lceil t/T_i\rceil-1$.

C_{int} 是处理中断的时间,包括保存处理中断并唤醒调度程序所必需的最小寄存器环境.

C_{sched} 是执行调度代码确定下一个要运行的任务所需的时间.比较就绪队列头部的任务和正在执行任务(活动任务)的优先级,并可能导致上下文的切换.

C_{timer} 是定时器中断中处理的系统定时器和保存处理中断并唤醒调度程序所必需的最小寄存器环境的时间.

C_{resume} 是抢占未发生时,返回到前一个活动任务所需的时间,包括恢复中断处理程序保存的寄存器环境,然后返回到任务的正常执行.

C_{store} 是将活动任务的状态保存到任务控制块 TCB(Task Control Block),并将这个任务控制块顺序插入到就绪队列中所需要的时间,插入操作的复杂度为 $O(n)$.

C_{load} 是从就绪队列中装入一个新的活动任务状态.一般来说,C_{load} 比 C_{store} 要小一些,因为后者要求一个到就绪队列的顺序插入.

C_{trap} 是处理任务正常结束时所生成的陷阱的时间,包括存储完成任务的 TCB 到开始队列中(存储已完成的任务),并选择就绪队列头作为下一个活动任务.

C_{pree} 和 C_{nonpree} 分别是抢占发生和未发生时的开销.

C_{exit} 是任务执行完毕,它的结束唤醒陷阱处理

程序并装入下一个活动任务所需的时间.

$C_{\mathrm{run}(i,u)}$是任务的装入开销,实时任务 τ_i 第 u 次周期性请求中断导致抢占行为的发生时为 C_{pree},否则为 C_{exit}.

$ChkP_{i,u}$是任务 τ_i 第 u 次执行的检查点有序集,其中的元素记录了本次执行期间与其直接相关的检查点,开始执行时刻减去 $C_{\mathrm{run}(i,u)}$ 和执行结束时刻分别作为集合的第 0 个元素和第 $2\cdot(w_{i,u}+1)+1$ 个元素.抢占集 $Pree_{i,u}$的每个元素提供了$ChkP_{i,u}$的两个元素,即$Pree_{i,u}$的第 k 个元素提供了$ChkP_{i,u}[2k+1]$和 $ChkP_{i,u}[2k+2]$,分别为 $Pree_{i,u}$的第 k 个元素标识的任务的本次执行请求中断时刻和执行结束的时刻,只有区间$[ChkP_{i,u}[2k],ChkP_{i,u}[2k+1]]$中发生的周期性任务请求会产生中断,其中 $i=0,1,\cdots,n-1,u=0,1,\cdots,\lceil t/T_i\rceil-1,k=0,1,\cdots,w_{i,u}$.

$C_{\mathrm{int}},C_{\mathrm{sched}},C_{\mathrm{resume}},C_{\mathrm{store}},C_{\mathrm{load}},C_{\mathrm{trap}},C_{\mathrm{pree}},C_{\mathrm{nonpree}}$,和 C_{exit}在文献[6]中给出了以下定义:

$$C_{\mathrm{pree}}=C_{\mathrm{int}}+C_{\mathrm{sched}}+C_{\mathrm{store}}+C_{\mathrm{load}},$$
$$C_{\mathrm{nonpree}}=C_{\mathrm{int}}+C_{\mathrm{sched}}+C_{\mathrm{resume}},$$
$$C_{\mathrm{exit}}=C_{\mathrm{trap}}+C_{\mathrm{load}}.$$

定理 1. 定时器驱动的 RM 调度机制下,实时任务 τ_i第 u 次执行的第 m 次被直接抢占的充分必要条件为 $DPISet_{i,u,m}\neq\varnothing$,其中

$$\begin{aligned}DPISet_{i,u,m}=\{\langle j,v\rangle\mid\exists j,v,k;&(k-1)\cdot T_{\mathrm{timer}}<Irq_{j,v}<k\cdot T_{\mathrm{timer}}=ChkP[2m+1]\wedge\\&ChkP_{i,u}[2m]\leq Irq_{j,v}\wedge j\in\{0,1,\cdots,i-1\}\wedge\\&v\in\{0,1,\cdots,\lfloor t/T_j\rfloor\}\wedge m\in\{0,1,\cdots,w_{i,u}\}\wedge\\&i\in\{1,2,\cdots,n-1\}\wedge u\in\{0,1,\cdots,\lfloor t/T_i\rfloor\}\wedge\\&k\in\{1,\cdots,\lfloor t/T_{\mathrm{timer}}\rfloor\}\}\end{aligned}\tag{1}$$

证明. 必要性.

在实时任务 τ_i第 u 次执行的开始时刻,集合 $ChkP_{i,u}$中只有两个有效元素,即 $ChkP_{i,u}[0]$ 和 $ChkP_{i,u}[1]$,集合 $Pree_{i,u}=\Phi$.

第一个元素:$ChkP_{i,u}[0]=r_{i,u}-C_{\mathrm{run}(i,u)}$,

最后一个元素:$ChkP_{i,u}[1]=r_{i,u}+C_i$.

每次抢占发生后,集合要在倒数第二个元素后面的位置插入两个元素,且最后一个元素的值要发生变化.元素 $ChkP_{i,u}[2m+2]$的值在抢占任务的本次执行结束能被确定,而最后一个元素在任务 τ_i第 u 次执行结束才能被确定.第 m 次抢占发生,集合 $ChkP_{i,u}$内容有如下变化:

$$ChkP_{i,u}[2m+1]=(\lfloor Irq_{j,v}/T_{\mathrm{timer}}\rfloor+1)\cdot T_{\mathrm{timer}},$$
$$ChkP_{i,u}[2(m+1)]=r_{Pree_{i,u}[m]}+e_{Pree_{i,u}[m]},$$
$$ChkP_{i,u}[2m+3]=r_{i,u}+e_{i,u}.$$

由于抢占发生,此时

$$r_{\mathrm{pree}_{i,u}[m]}=(\lfloor Irq_{j,v}/T_{\mathrm{timer}}\rfloor+1)\cdot T_{\mathrm{timer}}+C_{\mathrm{timer}}+C_{\mathrm{pree}}.$$

其中,$ChkP_{i,u}[2m+1]$和 $ChkP_{i,u}[2m+2]$是插入的两个元素,$ChkP_{i,u}[2m+3]$是当前集合 $ChkP_{i,u}$的最后一个元素.在采用定时器驱动的 RM 调度机制的实时嵌入式系统中,实时任务 τ_i第 u 次执行被任务 τ_j第 v 次执行请求中断的直接后继定时器中断时刻为$(\lfloor Irq_{j,v}/T_{\mathrm{timer}}\rfloor+1)\cdot T_{\mathrm{timer}}$对于任务 τ_i第 u 次执行,在这个中断时刻之前已经发生的非抢占中断的次数为 $SizeOf(NPIrq_{i,u})\mid_{(\lfloor Irq_{j,v}-r_{i,u})/T_{\mathrm{timer}}\rfloor+1)\cdot T_{\mathrm{timer}}}$,且定时器中断的次数为$(\lfloor(Irq_{j,v}-r_{i,u})/T_{\mathrm{timer}}\rfloor+1)$,因此,发生第 m 次抢占的定时器中断时刻前任务 τ_i第 u 次执行的历经时间为

$$\begin{aligned}e_{i,u}\mid_{k\cdot T_{\mathrm{timer}}}=&C_i+\sum_{k=0}^{m-1}e_{Pree_{i,u}[k]}+m\cdot(C_{\mathrm{pree}}+C_{\mathrm{exit}}+C_{\mathrm{timer}})+\\&SizeOf(NPIrq_{i,u})\mid_{(\lfloor Irq_{j,v}/T_{\mathrm{timer}}\rfloor+1)\cdot T_{\mathrm{timer}}}\cdot C_{\mathrm{nonpree}}+\\&\left(\left\lfloor\frac{Irq_{j,v}-r_{i,u}}{T_{\mathrm{timer}}}\right\rfloor-m+1\right)\cdot(C_{\mathrm{nonpree}}+C_{\mathrm{timer}})\end{aligned}\tag{2}$$

实时任务 τ_i第 u 次执行的第 m 次被抢占事件发生了,则 $\exists j,v$,即实时任务 τ_j第 v 次执行的请求必定发生在 $ChkP_{i,u}[m]$时刻之后,即

$$ChkP_{i,u}[2m]\leq Irq_{j,v}.$$

由于在定时器驱动的 RM 调度机制下,只有在直接后继的定时器中断时才启动调度程序,所以实时任务 τ_j第 v 次执行的请求要在这个定时器中断之前,且在实时任务 τ_i第 u 次执行的执行期间,即

$$\begin{aligned}\exists k((k-1)\cdot T_{\mathrm{timer}}\leq Irq_{j,v}<k\cdot T_{\mathrm{timer}}=&\\ChkP_{i,u}[2m+1]\leq r_{i,u}+e_{i,u}\mid_{k\cdot T_{\mathrm{timer}}}\wedge&\\k\in\{1,2,\cdots,\lfloor t/T_{\mathrm{timer}}\rfloor\}).&\end{aligned}$$

既然抢占发生,实时任务 τ_j比实时任务 τ_i的优先级高,即

$$j\in\{0,1,\cdots,i-1\}.$$

因此,$DPISet_{i,u,m}\neq\Phi$.

充分性.

由于 $DPISet_{i,u,m}\neq\Phi$,则 $\exists j,v,k((k-1)\cdot T_{\mathrm{timer}}\leq Irq_{j,v}<k\cdot T_{\mathrm{timer}}=ChkP_{i,u}[2m+1])$,即实时任务$\tau_j$第 v 次执行请求发生区间为$[ChkP_{i,u}[2m],k\cdot T_{\mathrm{timer}}]$,而实时任务 τ_i第 u 次执行在这个区间是获得处理器资源,且存在一个直接的后继定时器中断,这个定时器中断也发生在实时任务 τ_i第 u 次执行获得处理器资源的期间.

因又有 $j\in\{0,1,\cdots,i-1\}$,即实时任务 τ_j比实时任务 τ_i的优先级高,因此,在这个定时器中断中必定发生抢占,根据集合 $ChkP_{i,u}$中的有效元素个数可

知,这是实时任务 τ_i 第 u 次执行的第 m 次被抢占.

证毕.

依据定理 1,实时任务 τ_i 第 u 次执行的直接抢占集 $Pree_{i,u}$ 可由式(3)确定.

$$\{Pree_{i,u}[m]\}=\{\langle j,v\rangle \mid \forall j,v,x,y(\langle j,v\rangle,\langle x,y\rangle\in DPISet_{i,u,m}\wedge(j<x\vee(j=x\wedge Irq_{j,v}\leq Irq_{x,y})))\} \tag{3}$$

定理 2. 定时器驱动的 RM 调度机制下,在时间区间$[0,t]$内,实时任务集 T_{set} 在一种调度机制下是可调度的充分必要条件为式(4)成立.

$$\forall i,u(r_{i,u}+e_{i,u}<u\cdot T_i+D_i\wedge\tau_i\in T_{\text{set}}\wedge u\in\{0,1,\cdots,\lfloor t/T_i\rfloor\}) \tag{4}$$

证明. 必要性.

在时间区间$[0,t]$内,对于 $\forall i,u(\tau_i\in Tset\wedge u\in\{0,1,\cdots,\lfloor t/T_i\rfloor\})$,实时任务 τ_i 第 u 次执行获得处理器资源的时刻为 $r_{i,u}$;本次执行的信息由 3 个集合 $Pree_{i,u}$,$NPIrq_{i,u}$ 和 $ChkP_{i,u}$ 分别记录,可以得到本次的执行历经时间

$$e_{i,u}=C_i+\sum_{k=0}^{w_{i,u}-1}e_{Pree_{i,u}[k]}+w_{i,u}\cdot(C_{\text{pree}}+C_{\text{exit}}+C_{\text{timer}})+q_{i,u}\cdot C_{\text{nonpree}}+\left(\left\lceil\frac{t}{T_{\text{timer}}}\right\rceil-w_{i,u}\right)\cdot(C_{\text{nonpree}}+C_{\text{timer}}) \tag{5}$$

实时任务集 T_{set} 在时间区间$[0,t]$内由一种调度机制调度,能够满足可调度性,那么对于实时任务集中的任何实时任务 τ_i 在时间区间$[0,t]$内的任何一次执行都是可调度的,这次执行必定在相应的截止期 $u\cdot T_i+D_i$ 之前完成,即

$$\forall i,u(r_{i,u}+e_{i,u}<u\cdot T_i+D_i\wedge\tau_i\in T_{\text{set}}\wedge u\in\{0,1,\cdots,t/T_i\})\text{为真.}$$

充分性.

实时任务 τ_i 第 u 次执行获得处理器资源的时刻为 $r_{i,u}$,本次执行经历的时间为 $e_{i,u}$,本次执行结束的时刻为 $r_{i,u}+e_{i,u}$,又因为本次执行的截止期为 $u\cdot T_i+D_i$. 所以,式(4)为真,则实时任务集中的所有任务都是可调度的.

证毕.

定理 3. 定时器驱动的 RM 调度机制下,实时任务集 $\tau_0,\tau_1,\cdots,\tau_{n-1}$ 的相位集为 I,在时间区间$[0,t]$内,该任务集因抢占导致的额外开销为

$$O_{\text{preempt}}(I,t)=\sum_{i=1}^{n-1}\sum_{u=0}^{\left\lceil\frac{t}{P_i}\right\rceil-1}w_{i,u}\cdot(C_{\text{pree}}-C_{\text{nonpree}}) \tag{6}$$

证明.

实时任务 τ_i 第 u 次执行的直接抢占集 $Pree_{i,u}$ 中元素的个数 $w_{i,u}$ 为本次执行期间被直接抢占的次数,这个实时任务集在时间区间$[0,t]$内的总抢占次数为

$$N_{\text{pree}}(I,t)=\sum_{i=1}^{n-1}\sum_{u=0}^{\left\lceil\frac{t}{P_i}\right\rceil-1}w_{i,u}.$$

又,对于实时任务集中的每个任务,无论何时它的周期性请求发生,或者抢占正在运行的更低优先级任务;或者仅中断正在运行的更高优先级任务由调度程序将 TCB 顺序插入到就绪队列中,然后继续运行原来的活动任务,不发生抢占. 这两种情况的额外开销之差为$(C_{\text{pree}}-C_{\text{nonpree}})$. 在时间区间$[0,t]$中,抢占导致的额外开销为

$$O_{\text{preempt}}(I,t)=\sum_{i=1}^{n-1}\sum_{u=0}^{\left\lceil\frac{t}{P_i}\right\rceil-1}w_{i,u}\cdot(C_{\text{pree}}-C_{\text{nonpree}}).$$

证毕.

定理 4. 定时器驱动的 RM 调度机制下,实时任务集 $\tau_0,\tau_1,\cdots,\tau_{n-1}$ 的相位集为 I,在时间区间$[0,t]$内,该任务集的平均响应时间为

$$Rspn(I,t)=\frac{\sum_{i=1}^{n-1}\sum_{u=0}^{\left\lceil\frac{t}{P_i}\right\rceil-1}r_{i,u}+e_{i,u}-u\cdot T_i}{\sum_{i=1}^{n}\left\lceil\frac{t}{T_i}\right\rceil}. \tag{7}$$

证明.

实时任务 τ_i 第 u 次执行的原始请求时刻为该周期的开始时刻为 $u\cdot T_i$,由于相位值和其它更高优先级任务的执行,本次的实际启动时间为 $r_{i,u}$,执行结束的时刻为 $r_{i,u}+e_{i,u}$,则本次执行的响应时间为 $r_{i,u}+e_{i,u}-u\cdot T_i$. 那么,在时间区间$[0,t]$内,任务集的响应之和为

$$\sum_{i=1}^{n-1}\sum_{u=0}^{\left\lceil\frac{t}{P_i}\right\rceil-1}r_{i,u}+C_{\text{run}(i,u)}+e_{i,u}-u\cdot T_i.$$

在时间区间$[0,t]$内,任务集的执行次数之和为

$$\sum_{i=1}^{n}\left\lceil\frac{t}{T_i}\right\rceil.$$

则平均的响应时间为式(7).

证毕.

定理 1 和定理 2 从可操作的观点给出了每个任务单次执行级别上的直接抢占条件和可调度条件. 是对 Katcher 等人研究结果的进一步深化. 克服了 Katcher 模型在判断可调度性上过于保守的条件.

定理 3 和定理 4 给出了嵌入式系统实时性能的两个指标:抢占导致的额外开销和平均响应时间. 前者是刻划系统可调度利用率的一个指标,后者是软实时性能的量化指标.

定理 1～定理 4 构成了定时器驱动 RM 调度机

制下周期性任务的抢占模型 TDRMPModel，依据此模型，任务属性相位 I 是决定任务可调度性和抢占开销的因素之一. 由此，获得了可以延迟高优先级任务执行提高系统实时性能的方法，即相位可以作为优化变量，改变抢占关系，减少抢占导致的额外开销，进而提高系统的可调度利用率，改变系统的可调度性，降低系统的平均响应时间.

3 实时性能优化的方法

实时嵌入式系统因其资源的限制一般都运行在近乎满负荷的状态下，如果优化相位的计算由运行的目标系统完成是不可行的，那么如何获得优化的延迟并利用这些优化值?

受生物界寄生现象的启发，嵌入式系统仅保留能够利用优化相位值的利用机制，而如何产生或获取这些优化的可利用结果，则由离线运行在桌面计算机上的优化算法完成. 因此，这种实时性能优化的方法由两部分组成：嵌入式系统上的利用机制 AbsoPhasing和离线的获取机制 OptPhasing. 利用机制 AbsoPhasing 设计成为一组 API，将获取机制 OptPhasing 得到的优化相位值作为参数，设置实时任务的周期性请求. 由于这些操作仅发生在系统的初始化阶段，没有修改系统的调度机制，也不需要与外界交互，因此，当系统正常运行时，不会对原来的调度机制产生任何负面影响. 获取机制 OptPhasing 是寄生在桌面计算机上基于进化策略的优化算法，对嵌入式系统而言，以离线方式计算优化的相位值，这样就不会对目标系统的实时性能产生任何负面影响. OptPhasing 的核心内容是基于进化策略的优化算法.

由于利用机制 AbsoPhasing 仅是一组 API，下面只详细介绍获取机制 OptPhasing 中基于进化策略的优化算法 ESOptPhasing. 以硬实时系统为例，将抢占导致的额外开销和可调度性作为优化目标，所以，这个优化问题可以抽象地表达为如何选择一组 $\boldsymbol{I}=\{I_i \mid I_i \in R \wedge I_i \geq 0, i=0,1,\cdots,n-1\}$ 使得式(6)最小，且满足式(4)的条件，即如何在一个 n 维空间选择一个向量 $\boldsymbol{I} \in (R^+)^n$ 使得目标函数 $O_{\text{preempt}}(\boldsymbol{I},t)$ 最小并满足式(4)的约束条件，其中 $R^+=\{x \mid x \in R \wedge x \geq 0\}$. 由于这是一个带有约束条件的优化问题，采用罚函数的方法重新构造目标函数[9]. 结果如下所示：

$$O_{\text{pree}}(\boldsymbol{I},t)=\begin{cases} O_{\text{preempt}}(\boldsymbol{I},t), & \text{若式(4)是满足的} \\ t, & \text{其它} \end{cases} \tag{8}$$

获取机制 OptPhasing 中基于(1+1)——ES 进化策略[10]的优化算法 ESOptPhasing 所涉及的操作如下：

(1)编码与适应度函数

根据上面的问题描述，OptPhasing 中进化策略优化算法采用传统的实数编码以向量的形式表示个体基因，其适应度函数由待求解的目标函数式(8)转化而来.

实数编码的个体表示如下：

$$b^{t+1}=b^t+N(0,\sigma),$$

其中，$b^t \in (R^+)^n$ 标记为第 t 代的个体，$b^t=(b_0^t,b_1^t,\cdots,b_{n-1}^t)$，$(R^+)^n$ 为个体空间；$N(0,\sigma)$ 为服从正态分布的随机数，其均值为零，标准差为 σ.

适应度函数 $\Phi:(R^+)^n \rightarrow R^+$，$\Phi(b)=t-O_{\text{pree}}(b,t)$.

(2)变异

由于采用(1+1)——ES 进化策略，ESOptPhasing 中没有重组操作，进化操作只有突变一种，即

$$b_i^{t+1}=b_i^t+N_i(0,\sigma_i),$$

其中，$i=0,1,\cdots,n-1$.

(3)终止条件

进化过程中，每代都执行突变、计算适应度、选择等操作，不断反复使群体的素质得到改进，直至取得满意的结果. ESOptPhasing 的终止条件根据预先定义的最大进化代数和适应度的变化趋势两个判据，即有

终止条件 l：$(N,R^+)^n \rightarrow \{\text{True},\text{False}\}$，

其中，N 为自然数.

4 实验及结果分析

本文的实验是在仿真环境下进行的，所使用的系统属性的测量值来自一个基于 8 位微控制器的嵌入式系统. 系统属性和任务属性的测量方法见文献[11].

改善系统实时性能的优化方法是以硬实时系统为例，其优化目标是系统的抢占导致的额外开销和系统的可调度性，为了验证其有效性，特别是在改变系统可调度性上的有效性，随机生成 8 个负载较重的任务集，每个任务集由 5 个任务组成，即 $\tau_0,\tau_1,\cdots,\tau_4$，按优先级降序排列，周期、最坏执行时间(WCET)等相关属性见表 1. 为了确保每个任务

集在执行过程中任务之间的所有类型交互尽可能被考察到,将系统的运行时间 t 确定为各任务集中各任务的周期的最小公倍数,即任务集的超周期.对于每个任务集,将每个任务的最坏执行时间同时乘以一个因子,使系统的任务负载分别占系统负载的 20%,30%,…,100%,其中负载间隔为 0.1.这样,能够验证该实时性能优化方法在系统处于不同负载状态下的效果.

表 1　各实时任务集的属性

任务集	周期(T_0,T_1,T_2,T_3,T_4)(ms)	$WCET$(C_0.C_1,C_2,C_3,C_4) (ms)	超周期(ms)
0	(0.8, 1.5, 3.0, 4.0, 6.0)	(0.087, 0.231, 0.365, 0.683, 1.056)	12.0
1	(0.75, 2.4, 6.0, 8.0, 10.0)	(0.074, 0.451, 1.568, 1.071, 5.239)	120.0
2	(0.9, 4.5, 7.5, 12.0, 15.0)	(0.043, 1.077, 1.366, 2.397, 3.344)	180.0
3	(0.7, 1.5, 2.4, 3.0, 6.0)	(0.052, 0.083, 0.214, 0.356, 1.002)	84.0
4	(2.5, 4.0, 5.0, 7.5, 15.0)	(0.159, 0.218, 0.603, 2.582, 4.651)	60.0
5	(3.0, 4.5, 5.0, 6.0, 9.0)	(0.217, 0.144, 0.228, 1.054, 2.469)	90.0
6	(1.0, 2.0, 2.5, 6.0, 10.0)	(0.066, 0.167, 0.550, 1.263, 2.040)	60.0
7	(1.5, 2.5, 6.0, 12.0, 15.0)	(0.158, 0.192, 1.243, 1.360, 3.595)	60.0

测量得到的三个相关系统属性值如下:

$C_{\text{pree}}=52.875\mu s$,　$C_{\text{nonpree}}=33.345\mu s$,

$C_{\text{exit}}=33.333\mu s$,　$T_{\text{timer}}=200\mu s$.

对于每个任务集生成的不同任务负载,AbsoPhasing 分别使用 OptPhasing 优化得到的相位值和传统的相位值,即(0,0,0,0,0),设置任务的周期性请求.通过仿真实验得到的优化前后各任务集由抢占导致的额外开销数据组成,两者比较如图 2 所示.

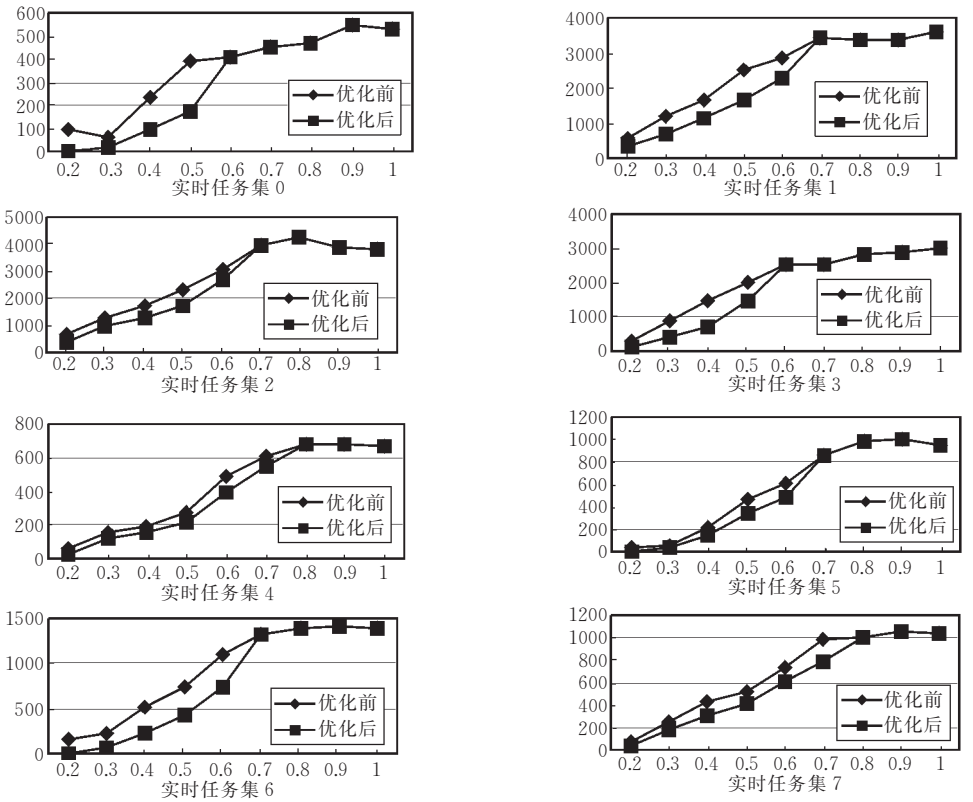


图 2　在系统不同任务负载状态下优化前后各实时任务集抢占导致额外开销的比较

实验数据显示该优化方法对于各任务集的不同任务负载优化的效果不同,可优化任务负载上限如表 2 所示.由于实验选择的定时器周期是 200μs,在每个周期如果有抢占发生 26.44%的 CPU 时间作为抢占开销,即使是不发生抢占定时器的开销也占用 16.67%的 CPU,所以每个任务集生成的任务负载高于 1−16.67%=83.33%就不可调度.又因为在一个定时器周期中还有其它的开销,如任务请求中断等,因此可调度的任务会更低.实验结果表明优化前后系统的任务负载最高为 0.7,最低为 0.5,由于实验选择的任务负载递增的增量为0.1,所以这个上限会低于真正的上限.在任务负载上限

处,该优化方法由抢占导致的减少的额外开销为9.68%～55.0%.

表 2 任务集的优化前后任务负载上限和可调度利用率

任务集	任务负载上限		可调度利用率(%)	
	优化前	优化后	优化前	优化后
0	0.5	0.5	58.76	65.40
1	0.5	0.6	59.96	62.20
2	0.6	0.6	62.81	67.80
3	0.5	0.5	59.20	62.00
4	0.7	0.7	75.55	75.90
5	0.6	0.6	68.57	69.24
6	0.6	0.6	64.85	65.10
7	0.6	0.7	68.81	71.30

从图 1 和表 2 中的数据可知,该优化方法在系统负荷未过载的情况下都有效,在任务负载较轻时可以将由抢占导致的额外开销减少到 0,如图 1 中的任务集 0、任务集 5 和任务集 6 的 0.2 任务负载.在优化前系统轻微过载时,该优化方法能够通过调整任务的相位减少抢占导致的额外开销,从而改变该任务负载的可调度性,如表 2 中任务集 1 的 0.6 任务负载和任务集 7 的 0.7 任务负载.表 2 的数据还表明,该方法能够使该 RM 调度机制的可调度利用率提高 0.25%～6.64%.

5 结 论

定时器驱动的 RM 调度机制下周期性任务抢占模型 TDRMPModel 是对 Katcher 等人对于该调度机制模型的进一步完善,明确了实时周期性任务直接抢占发生的充分必要条件,使得可调度性的判定条件成为充分必要条件,进而确定了抢占导致的额外开销和系统的平均响应时间.根据抢占模型 TDRMPModel 中相位对整个系统实时性能的影响,受生物界寄生现象的启发,将利用机制和获取机制分离.获取机制利用寄生于桌面计算机上的基于进化策略的优化算法以离线方式计算优化的相位值.利用机制使用优化的相位值在系统的初始化阶段完成任务的周期性请求设置.因此,无论是包含复杂计算的获取机制,还是简单设置的利用机制,都不会在系统进入正常运行阶段对其实时性能产生负面影响.以硬实时系统为例的实验结果表明,本文依据抢占模型提出的方法适用于使系统未过载或轻微过载的任务集.对于这类系统,此方法能够使抢占额外开销减少 9.68～55.0%;可调度利用率提高 0.25～6.64%;但是最大的贡献在于,不采用其它优化方法,如优化源代码、增加系统的计算能力等,就能够改变一个轻微过载系统的可调度性.

本文建立的模型和提出的方法面向仅有周期性任务运行的嵌入式系统,这种理想的限定条件,限制了它们在工程实际中的应用.加入非周期性任务的调度算法,建立包含周期性任务和非周期性任务的抢占模型是下一步的研究工作.精确化本文优化方法的适用条件是下一步的另一项研究工作.

参 考 文 献

1 Liu C. L., Layland J. W.. Scheduling algorithms for multiprogramming in a hard-real-time environment. Journal of the ACM, 1973, 30(1): 46～61

2 Bernat G., Colin A., Petters S. M.. WCET analysis of probabilistic hard real-time systems. In: Proceedings of the 23rd Real-Time Systems Symposium, Austin, Texas, USA, 2002, 279～288

3 Tan T. K., Raghunathan A., Jha N. K.. Embedded operating system energy analysis and macro-modeling. In: Proceedings of International Conference Computer Design, Freiburg, Germany, 2002, 515～522

4 Stefan M. P.. Comparison of trace generation methods for measurement based WCET analysis. In: Proceedings of the 3nd International Workshop on Worst Case Execution Time Analysis. Porto, Portugal, 2003, 61～74

5 Wang Y., Saksena M.. Scheduling fixed-priority tasks with preemption threshold. In: Proceedings of IEEE Real Time Computing Systems and Applications Symposium, Hong Kong, China, 1999, 328～335

6 Katcher D., Arakawa H., Strosnider J.. Engineering and analysis of fixed priority schedulers. IEEE Transactions on Software Engineering, 1993, 19(9): 920～934

7 Lisper B.. Fully automatic, parametric worst-case execution time analysis. In: Proceedings of the 3nd International Workshop on Worst Case Execution Time Analysis. Porto, Portugal, 2003, 85～88

8 Dick R. P., Lakshminarayana G., Raghunathan A., Jha N. K.. Power analysis of embedded operating systems. In: Proceedings of Design Automation Conference, Los Angeles, CA, 2000, 312～315

9 Michalewica Z., Janikow C.. Handling constraints in genetic algorithms. In: Proceedings of the 4th International Conference on Genetic Algorithms, Morgan Kaufmann, 1991, 151～157

10 Bäck T.. Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms. New York: Oxford University Press, 1996

11 Wang Ji-Yong, Zhao Hai, Lin Tao, Wang Jin-Dong, Han Guang-Jie. A software-only method for measuring execution times. Chinese Journal of Electronics, 2004, 13(3): 399～403

WANG Ji-Yong, born in 1975, Ph. D. candidate. His research interests include embedded systems and embedded Internet.

ZHAO Hai, born in 1959, professor and Ph. D. supervisor. His research interests focus on embedded Internet and information fusion.

LIN Tao, born in 1975, Ph. D. candidate. His research interests focus on embedded Internet.

WANG Xiao-Ying, born in 1975, Ph. D. candidate. Her research interests focus on embedded Internet.

WANG Jin-Dong, born in 1976, Ph. D. candidate. His research interests focus on embedded Internet.

HAN Guang-Jie, born in 1972, Ph. D. candidate. His research interests focus on embedded Internet.

Background

Design and implementation of embedded systems are fundamentally restricted in their resources. In despite of the great progress in the art of hardware technologies, because of cost and other factors, resources of embedded systems are more and more lacking with increasing function needs of the target environment. In the initial phase of their implementations, a lot of embedded systems are not performing to specifications because they have very limited resources. Therefore, it is essential to optimize real-time performance of resource-constrained embedded systems.

For many embedded systems, the underlying preemptive scheduling mechanism is an overly stringent requirement, which restricts to execute high priority tasks, resulting in lower source utilization and poor average performance. Taking the idea of behaviors and way of parasites in the nature, this paper presents the parasitism optimization idea and its implementation method, by which an embedded system as parasite can uses nutrition resources produced by the desktop computer system called host. Based on system theory, properly postpones releasing high priority tasks to overcome the deficiency of overall performance decline due to local optimization of high priority tasks.

It is the successive research of the project supported by the National Natural Science Foundation of China, under grant No. 69873007. The project is titled "The Research of Embedded Internet Architecture and Application". Its objective is to study the architecture for connecting fieldbus-connected devices to Internet. This research involves in embedded RTOS, embedded TCP/IP protocol stack, and transmission between fieldbus protocols and TCP/IP protocols. As respect to embedded RTOS, its work is focused on optimizing real-time performance of embedded systems. The study group has presented several ideas to measure and optimize real-time performance of embedded systems under the preemptive scheduling mechanisms, such as event-driven RM, event-driven EDF.