文章编号:1001 - 9081(2009)06 - 1509 - 05

开放式实时系统双层调度框架的一种改进方案

金永贤,黄经州
(浙江师范大学 数理与信息工程学院,浙江 金华 321004)
(jyx@ zjnu. cn)

摘 要:分析了开放式实时系统双层调度框架的调度特点,指出了其仅适用于硬实时环境的缺陷。在保证硬实时应用可调度性的基础上,针对硬实时和软实时应用的不同特点提出了一种改进方案,增加了软实时应用的处理,拓展了原方案的适用范围,完善了开放式实时系统的双层调度框架,最后用理论证明与仿真实验验证了改进方案的有效性。

关键词:开放式实时系统;调度框架;可调度性;软实时;硬实时
中图分类号: TP316.2 **文献标志码**:A

Improved two-level scheduling framework in open real-time system

JIN Yong-xian, HUANG Jing-zhou
(*College of Mathematics, Physics and Information Engineering, Zhejiang Normal University, Jinhua Zhejiang* 321004, *China*)

Abstract: An improved two-level scheduling framework based on different characteristics of hard and soft real-time applications was proposed through analyzing the two-level scheduling framework in the open system. This improved approach can expand the application range through adding the process of dealing with soft real-time applications. The simulation results show its validity.

Key words: open real-time system; scheduling framework; schedulability; soft real-time; hard real-time

0 引言

随着实时系统应用的发展,硬实时、软实时和非实时应用并存的开放式实时系统应用越来越普遍,这给系统调度机制带来了新的需求和挑战,以前针对封闭式实时系统提出的单一调度方法已经不能满足这种需求。于是,近几年人们提出了开放式实时系统的概念。开放式实时系统的主要特点是:独立开发和验证的各个实时应用或非实时应用可在运行时动态配置和加入系统,与系统中原有的应用并发运行,并且当系统进行动态扩展时,不需要做全局的可调度性分析[1]。

实时系统系统调度策略层面的问题是关系到时间约束能否得到满足的关键问题之一。目前,关于开放式实时系统调度策略主要有两类:第一类是文献[1]提出的层次式调度框架内基于服务器集成多种调度策略的方法。它是基于 CUS (Constant Utilization Server)和 TBS(Total Bandwidth Server)的预留带宽算法,并以此为基础构建了双层调度框架,它强调面向应用系统个性化任务调度,不同的约束采用不同的系统调度策略;第二类是文献[2]提出的统一架构的多调度策略融合方法。强调多种调度策略的统一系统调度模型,采用统一的结构实现多类调度策略的配置,但系统中只存在一种调度策略,它强调调度策略配置的灵活性。而从现有的成果看,开放式实时系统调度方法大部分采用文献[1]提出的基于服务器的策略。

文献[3]中的 GPS(Generalized Processor Sharing)算法将实时应用理想化为粒度可无限细分的工作流,然后每个实时任务根据需要,分配一定的处理机带宽。文献[4]中的 EGPS (Earliest Completion-Time GPS)算法继承了[3]中的思想,它先计算在 GPS 下各个任务到达后的完成时间,然后再按完成时间越小越优先的顺序执行系统中的全部活动任务。文献[5 - 6]中的 CBS(Constant Bandwidth Server)与文献[7]中的 H-CBS(Hierarchical CBS)算法通过资源预留机制在单个系统中集成了硬实时任务和软实时多媒体任务,且它们只能在被分配的带宽中执行,不会对其他实时任务造成影响,做到了带宽隔离。文献[8]提出了两个算法 EDL-RTO (Earliest Deadline as Late as possible-Red Tasks Only)和 EDL-BWP(Blue When Possible),目的是为了使软实时非周期请求的平均响应时间最小化,同时使周期任务的 QOS 不低于某个指定的值。文献[9]针对软实时多媒体应用的系统提出了一种不可抢占的 gEDF(group Earliest Deadline First)算法,并通过实验指出这种算法在处理软实时多媒体应用时更加高效。文献[10]致力于解决多处理器平台下软实时系统的调度,指出相对于在每个处理器上分别应用 EDF 算法,在多处理器平台上应用 global EDF 算法将使软实时任务获得更高的处理器利用率。文献[11]提出的 PShED (Processor Sharing with Earliest Deadlines First)算法提供了调度任务之间的独立性。文献[12]提出的 RPDS(Rigorously Proportional Dispatching Server)算法建立了一种新的层次式调度框架,它以时间片为单位对各类任务进行调度。文献[13]提出了 OARtS(Open Adaptive Real-Time Scheduling)框架,将自动控制的思想引入了开放式实时系统调度,它可以根据系统资源的情况动态调节任务的实时等级。但它并未对任务的某些特性参数如是否含有不可抢占区、是否使用全局资源等做出讨论,且它们需要大量的计算,增加了系统的负担。文献[14]提出二维优先级实时调度机制,对调度算法划分优先级并分配相应带宽,但其带宽在系

收稿日期:2008 - 11 - 24;**修回日期**:2009 - 03 - 27。

作者简介:金永贤(1964 -),男,浙江东阳人,副教授,主要研究方向:实时与嵌入式系统、蓝牙技术; 黄经州(1982 -),男,山东泰安人,硕士研究生,主要研究方向:实时系统。

统运行过程中无法动态调整,系统资源很难得到充分利用。

上述这些研究对任务类型(周期任务、非周期任务等),任务特性(是否含有不可抢占区、是否会请求全局资源)等都没有进行综合考虑与讨论。相比之下,文献[1]提出的基于TBS和CUS服务器的层次式调度策略考虑了开放式实时系统的各个方面,可以容纳实时与非实时应用、多特性任务,可以对复杂开放式实时环境下的实时与非实时应用进行调度,在开放式实时系统调度中有其优越性。本文着重研究了文献[1]提出的实时调度策略,指出此方案将硬实时与软实时应用作为同一类实时应用进行无区分调度,无法保证服务质量,并提出了一种改进方案。新方案加入了软实时应用的处理,解决了硬实时与软实时应用无区分调度问题,并考虑到了实时应用所含的不可抢占区对调度带来的影响,使之适应性更强,应用范围更广,并拓展了开放式实时系统层次式调度理论与方法。

1 开放式实时系统双层调度框架分析与改进

1.1 相关概念

定义1 任务。指完成某一特定功能的软件实体,它是实时调度的一个基本单位。任务在其生命期的某一次执行称为该任务的一个作业。由多个任务组成的任务集就称为应用。

定义2 服务器和服务器速率。在本文中服务器指系统调度机制创建的特殊任务,它为调度对象提供服务[15]。系统中包含多个服务器,每个服务器相当于一个慢速的处理机。设系统处理机的速度为1,将服务器看作一个速度为 σ_k 的虚拟处理机,应用在这个虚拟处理机上是恰好可调度的,那么这个虚拟处理机的速度与系统处理机速度的比值就称为服务器速率。服务器 S_k 的速率 $\sigma_k < 1$。

定义3 应用事件。指某个时刻在应用内部发生的动作。为了准确补充服务器的预算并设定其截止期,从而保证系统的可调度性,服务器调度器必须跟踪应用事件的发生时刻。应用事件包括:1) 应用 A_k 中的作业释放或者完成;2) A_k 中的作业请求或者释放全局资源;3) A_k 中的作业进入或者离开不可抢占区。

1.2 双层调度框架分析及改进的预期目标

图1是开放式实时系统双层调度框架,此框架所能处理的应用类型为实时和非实时,其调度策略的详细情况以及有关非实时应用调度的处理参见文献[16],在此不再赘述。对于实时应用的调度,原框架有如下特点:

1)每个实时应用拥有自己的调度算法和服务器,每个服务器占用一定的处理机带宽,维持所在应用的任务队列并参与系统层调度。

2)实时应用中所包含的任务类型不限,周期任务、非周期任务、偶发任务均可。

3)没有对任务是否可预测、是否可相互抢占、是否使用全局资源、是否含有不可抢占区等特性参数做严格限定,但对于不可预测、使用全局资源或者含有不可抢占区的任务,必须为它们预留足够的带宽以保证其可调度性。

4)每个实时应用在进入系统之前不必作全局可调度性分析,但要经过准入测试,满足条件的实时应用才能进入系统执行。

通过上述分析可知,此调度框架可以处理具有各种特性的多类型实时任务,并可保证它们的可调度性,但是,它对实时应用的处理没有考虑到硬实时和软实时应用的区别,也就是说,原调度框架只适用于硬实时应用。如果 T_i 是硬实时任务,则 e_i 就是作业在最坏情况下无中断的执行时间;如果 T_i 是软实时任务或者最大努力任务,则 e_i 即为一个统计值或者估计值[12]。那么对于由软实时任务组成的软实时应用,它执行时所需要的处理机速率以及处理时间也是一个估计值。如果采用原调度策略处理软实时应用,若软实时应用的实际执行时间大于估计值,系统的可调度性就无法得到保证。另外,硬实时和软实时应用对截止期的要求也不同:硬实时应用不可错过截止期,否则会造成严重后果;软实时应用有一个估计的截止期,即使某些应用错过了截止期也不会造成重大损失。但在实际应用中,我们应该尽量降低其截止期错失率,以提供更好的服务质量。

为此,本文试图在保持原调度框架开放性的基础上,对其中实时应用的调度部分做出适当改动,并提出新的调度方法以实现以下目标:

1)体现硬实时应用与软实时应用在实际调度中的优先级差异。

2)保证硬实时应用的可调度性,使其截止期错失率仍为0。

3)使软实时应用总体的截止期错失率小于1。

4)非实时应用不限定其截止期,但所采用的调度算法应尽量防止"饿死"情况的出现。

1.3 改进方案及调度算法

为实现上述提出的目标,本文对原系统架构做如下改动:将原先OS层的服务器就绪队列分为三部分:硬实时、软实时和非实时(其中,非实时队列只含有一个服务器)。硬实时与软实时就绪队列均采用EDF调度算法,非实时应用就绪队列采用Time-Sharing或者FCFS调度算法。改进后的调度框架如图2所示,本文提出的算法将在此基础上进行描述。为了便于讨论,做如下设定:

1)硬实时和软实时应用中的任务均为周期性任务,它们

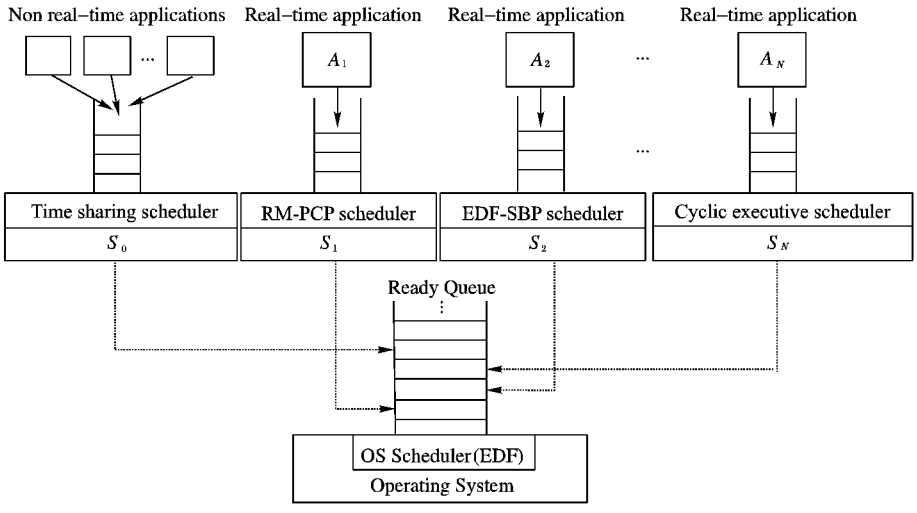

图1 开放式实时系统双层调度框架

均采用可抢占的调度算法。

2)各应用都是可预测的,它们所包含任务的周期等于相对截止期,且作业均在周期开始时刻释放。

3)允许硬实时和软实时应用含有不可抢占区或者使用全局资源,如果非实时应用中含有不可抢占区或者使用全局资源,那么将非实时应用作为软实时应用参与系统调度。

4)本文中采用的基本调度单位为任务,应用的可调度性通过它所含任务的可调度性来证明,即若应用中所含任务都是可调度的,那么应用也是可调度的。

新算法的整体思想为硬实时、软实时和非实时三个服务器就绪队列赋予不同的优先级:当系统中有硬实时作业处于就绪状态时,优先执行硬实时作业;当硬实时就绪队列为空时,执行软实时作业;当上述两个就绪队列均为空时,执行非实时作业。若系统中正在执行低优先级作业时有高优先级作业变为就绪态,高优先级作业立即抢占低优先级作业。

下文中各个参数定义为:

U_0 — 系统开始运行时分配给非实时应用的固定带宽,系统结束前其值不发生改变。

U_h — 系统中所有硬实时应用的总 CPU 利用率。

U_s — 系统中所有软实时应用的总 CPU 利用率。

U_t — 系统当前总的CPU利用率,即已分配的系统总带宽值,它满足 $U_t = U_h + U_s + U_0$ 且 $U_t \leqslant 1$。

B_j — 除作业 J_j 外其他所有作业中不可抢占区持续时间的最大值。

应用进入系统前要经过准入测试环节,非实时应用中若含有不可抢占区或者需要使用全局资源,那么它将升级为软实时应用,否则直接进入系统加入非实时应用调度队列。当一个新的实时应用 A_k 请求进入系统时,它必须提供下列信息:

1)A_k 所使用的调度算法 Σ_k。

2)A_k 单独在慢速处理机上执行且恰好可调度情况下所需要的处理机速率 σ_k。

3)若 A_k 中的任务含有不可抢占区或者使用全局资源,给出其所有不可抢占区或者使用全局资源的最大值 L_k。

4)若 A_k 为优先级驱动,给出 A_k 中所有作业相对截止期的最小值 δ_j;若 A_k 为时间驱动,给出 A_k 中两个连续事件发生间隔的最小值 δ_j[1]。

然后,系统将根据所得信息做以下响应。

1)确定应用 A_k 的服务器类型为CUS或者TBS(本文中所采用的服务器类型均为 TBS,这两种服务器区别参见文献[16])。

2)为应用 A_k 分配处理机带宽 U_k(本文中应用是可预测的,$U_k = \sigma_k$)。

3)若 $U_t + U_k + \max\limits_{1 \leqslant j \leqslant N}\{B_j/\delta_j\} > 1$(其中 N 为包括应用 A_k 在内的系统中实时应用的总数),拒绝 A_k;否则,接受 A_k 并做以下工作:为应用 A_k 创建一个速率为 U_k 的 TBS 服务器 S_k;将 S_k 的预算与截止期均设为 0,并将 U_t 的值增加 U_k[1]。

在应用层,若应用有新作业释放,则依照调度器所采用的调度算法按顺序插入作业就绪队列。在系统层,系统需要不断地检查各实时应用的作业就绪队列。当有作业处于就绪状态,系统补充应用所在服务器的预算,设定截止期,并将其加入对应的服务器就绪队列。服务器调度算法描述如下:

1)检查硬实时服务器就绪队列是否为空,若空,转 2);否则调度队列中的第一个服务器,执行服务器内的作业。该服务器调度完成后,将其移出服务器就绪队列,转 1)。

2)检查软实时服务器就绪队列是否为空,若空,转 3);否则调度队列中的第一个服务器,执行服务器内的作业。该服务器调度完成后,将其移出服务器就绪队列,转 1)。

3)检查非实时服务器是否处于就绪态,若是,执行服务器内的作业至下一个应用事件发生时刻,转 1)。

2　改进方案可调度性分析证明

下面将证明改进后的调度方法和算法是否能达到预期的目标。

1)通过算法描述可以看出,硬实时应用所释放的作业永远优于软实时作业或者非实时作业执行,硬实时作业被阻塞的情况仅在软实时作业在其不可抢占区执行或者使用全局资源时发生。那么,若有上述情况发生,硬实时作业是否能保证在截止期之前完成呢? 换句话说,软实时应用是否会影响硬实时应用的可调度性呢? 这个问题将在第 2)点中证明。

2)上述调度方案能否保证硬实时应用的截止期错失率为 0,即保证其所含作业必须均在截止期之前完成。

为此需要证明三点:a)若实时应用中不含不可抢占区或者使用全局资源,硬实时应用是可调度的。b)硬实时应用中所含的不可抢占区或者使用全局资源不影响其他硬实时应用的可调度性。c)软实时应用中所含的不可抢占区或者使用全局资源不影响硬实时应用的可调度性。通过下述三个定理证明。

定理 1　采用本文的调度框架,若系统采用 EDF 算法调度硬实时服务器就绪队列,当实时应用中不含不可抢占区或者使用全局资源,硬实时应用是可调度的。

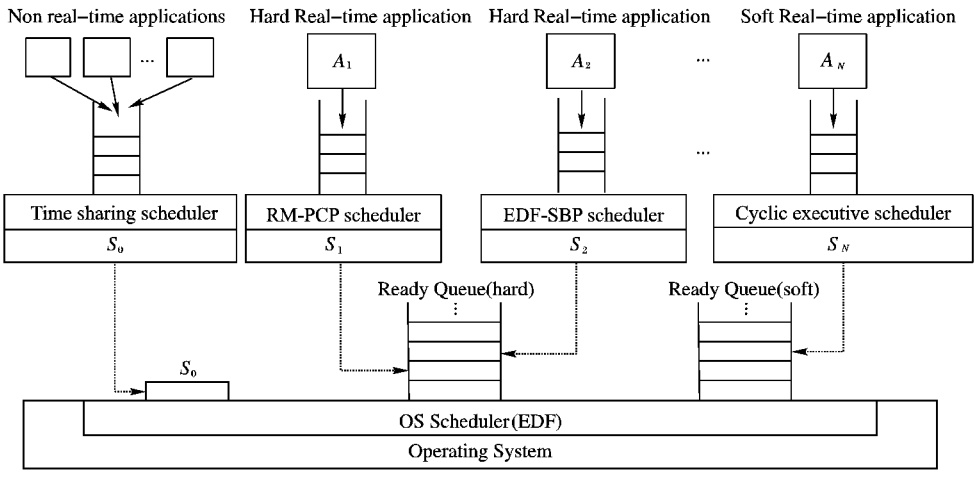

图 2　开放式实时系统调度框架的改进方案

证明

a)实时应用包含硬实时应用和软实时应用,当它们不含不可抢占区或者使用全局资源时,依据本文算法,软实时作业在任何时候都无法对硬实时作业形成阻塞,因此软实时应用对硬实时应用的可调度性没有影响。

b)文献[17]证明了在含有多个硬实时任务的任务

集合中,当且仅当所有任务的CPU利用率之和不超过1,此任务集合在EDF算法下是可调度的,若任务集合可以被其他算法调度,那么也可以用EDF算法来调度。从这个意义上讲,EDF算法是最优的。在本文中,所有硬实时应用的CPU利用率之和 $U_h \leqslant U_t \leqslant 1$,满足上述充要条件,因此硬实时应用是可调度的。 证毕

定理2 采用本文的调度框架,若系统采用EDF算法调度硬实时服务器就绪队列,硬实时应用中所含的不可抢占区或者使用全局资源不影响其他硬实时应用的可调度性。

证明 采用反证法。假设任务 T_1 中含有的不可抢占区影响了任务 T_2 的可调度性,它们释放的两个作业分别为 J_1、J_2,J_1、J_2 的释放时刻分别为 r_1、r_2,截止期分别为 d_1、d_2,作业 J_1 中不可抢占区的执行区间为 $[t_0,t_1]$,作业 J_2 的执行时间为 e_2,任务 T_2 的周期为 p_2,如图3所示。

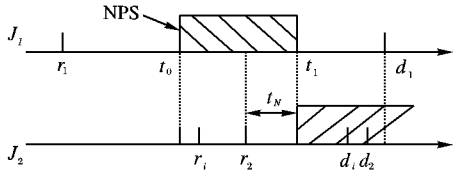

图3 NPS(不可抢占区)与可调度性例证

1) 因 J_1 中的不可抢占区会对 J_2 造成阻塞,那么一定有 $d_2 < d_1$,且 $t_0 \leqslant r_2 < t_1$。假设 r_2 与 t_1 之间的时间长度为 t_N,即作业 J_2 释放后要等待 t_N 才能得到执行,此时作业 J_2 将错过其截止期,则有:

$$t_N + e_2 > d_2 - r_2 = p_2 \tag{1}$$

成立,整理得:

$$1 < \sigma_2 + t_N/p_2 < U_t + \max_{1\leqslant j\leqslant N}\{B_j/\delta_j\} \tag{2}$$

与准入测试条件矛盾,假设错误。也就是说,即使某硬实时作业被不可抢占区阻塞,在满足准入测试的条件下,它也是可调度的。

2) 考虑一种更复杂的情形:假设有一个作业 J_i,周期为 p_i,执行时间为 e_i,其释放时刻 r_i 在 t_0 之后而截止期 d_i 位于 d_2 之前,且它的存在影响了 J_2 的可调度性。由1)中得知,J_i 是可调度的。根据作业 J_i、J_2 的周期 p_i、p_2 之间的关系,分以下两种情况讨论:

a) $p_i \leqslant p_2$。

根据条件可知:$(e_i/p_2) \leqslant (e_i/p_i)$,而由图3可知,不论 J_i 的释放时刻位于何处,均有:

$$t_N + e_i + e_2 > p_2 \tag{3}$$

成立,因此:

$$1 < (t_N/p_2) + (e_i/p_2) + \sigma_2 \tag{4}$$

进一步,

$$1 \leqslant (t_N/p_2) + (e_i/p_i) + \sigma_2 = (t_N/p_2) + \sigma_i + \sigma_2 < U_t + \max_{1\leqslant j\leqslant N}\{(B_j/\delta_j)\} \tag{5}$$

与准入测试条件矛盾,假设错误。

b) $p_i > p_2$。

由假设可知 $t_0 < r_i < r_2$,则:

$$\frac{(t_1 - t_0)}{p_2} + \frac{e_2}{p_2} + \frac{e_i}{p_i} \geqslant \frac{(t_1 - r_i)}{p_2} + \frac{e_2}{p_2} + \frac{e_i}{p_i} \geqslant \frac{(t_1 - r_i)}{p_2} + \frac{(e_2 + e_i)}{p_i} \geqslant \frac{(t_1 - r_i + e_2 + e_i)}{p_i} > \frac{(t_1 - r_i + d_i - t_1)}{p_i} = 1 \tag{6}$$

另一方面,

$$\frac{(t_1 - t_0)}{p_2} + \frac{e_2}{p_2} + \frac{e_i}{p_i} \leqslant \max_{1\leqslant j\leqslant N}\{(B_j/\delta_j)\} + \sigma_2 + \sigma_i < \max_{1\leqslant j\leqslant N}\{(B_j/\delta_j)\} + U_t \tag{7}$$

于是有:

$$1 < \frac{(t_1 - t_0)}{p_2} + \frac{e_2}{p_2} + \frac{e_i}{p_i} < \max_{1\leqslant j\leqslant N}\{(B_j/\delta_j)\} + U_t \tag{8}$$

即:

$$1 < \max_{1\leqslant j\leqslant N}\{(B_j/\delta_j)\} + U_t \tag{9}$$

与准入测试条件矛盾,假设错误。

从a)和b)知,作业 J_i 并不影响 J_2 的可调度性。其他情形也可以采用上述方法证明。

综合1)和2),可以知道,无论直接还是间接情况下,硬实时作业中的不可抢占区或者使用全局资源都不会影响其他硬实时作业的可调度性,而由硬实时任务组成的硬实时应用的可调度性也可以得到保证。 证毕

定理3 采用本文的调度框架,若系统采用EDF算法调度硬实时服务器就绪队列,软实时应用中所含的不可抢占区或者使用全局资源不影响硬实时应用的可调度性。

证明 本文中,算式 $\max_{1\leqslant j\leqslant N}\{(B_j/\delta_j)\}$ 针对所有的硬实时和软实时应用,当进入不可抢占区执行时,软实时作业与硬实时作业对硬实时作业可调度性的影响并无不同。因此,本定理证明过程与定理2相同。 证毕

通过上述三定理可以得知,本文所采用的算法能够保证硬实时应用的可调度性。

3) 采用本调度框架,软实时应用总体的截止期错失率是否小于1。

若系统中不存在硬实时应用,那么软实时应用将获得最高优先级。通过定理1和定理2得知,在统计意义上,它们是可调度的。因此,造成软实时应用错过截止期的原因就是因为硬实时应用的存在。设想系统中存在 m 个硬实时任务和 n 个软实时任务,这 $m+n$ 个实时任务周期的最小公倍数为 p。若 n 个软实时任务的每次执行都错过了它们的截止期,那么依据本文算法,在时间段 p 内,非实时任务不可能得到执行。也就是说,这个时间段内,所有任务可调度所需要的CPU利用率超过了1,而所有任务都是周期性的,这种情况不会得到改善,与准入测试条件矛盾。所以,时间段 p 内必然有软实时作业在截止期内完成,也可以推断软实时应用总体的截止期错失率小于1。

4) 系统为非实时应用预留了一部分带宽 U_0,因为双层调度框架可以做到带宽隔离,因此在时间段 p 内(沿用3)中的设定),非实时作业可以得到 $p \times U_0$ 时间执行。对此作者已在文献[16]中作出详细讨论,可以很好防止"饿死"状况的出现,且能提高其服务质量。

3 仿真实验及分析

为检验本文中所提出的改进调度方案的有效性,对调度算法进行了仿真。因算法针对实时应用提出,所以在仿真过程中仅选择了硬实时和软实时任务进行测试。仿真过程旨在检验是否达到了本文提出的调度目标中的1)~3),为了尽量减少其他应用及调度开销对仿真过程产生的影响,将实时任务的周期设置较长,在10~60 s之间,其周期、所需带宽和不

可抢占区等任务属性均由系统随机产生,且系统始终在满负荷状态下运行。实验产生结果以任务执行的次数做统计,如图 4 所示。从图上可以看出,在系统满负荷条件下,多个软实时和硬实时任务各执行了 150 次,硬实时任务无一错过截止期(截止期错失率曲线与 X 轴重合),而软实时任务截止期错失率的峰值也仅有 5%(前 30 个任务截止期错失率为 0),仿真结果表明改进框架及算法达到了预期目标。

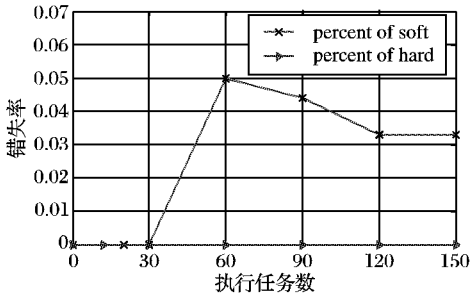

图 4　硬、软实时任务执行结果对比

与原方案相比,改进方案虽然不能保证软实时任务都在截止期内完成,然而在统计意义上,系统为软实时任务提供了更好的并发性,只要保证软实时任务中的不可抢占区不影响硬实时任务可调度性,系统就可以接受更多的软实时任务并发执行,使系统始终处于满负荷状态,充分利用系统资源。

4　结语

本文深入研究了开放式实时系统双层调度策略,指出了其仅适应硬实时环境的局限性,在保证硬实时可调度的基础上,加入了软实时应用的处理,提出了一种改进方案,解决了原方案将硬实时与软实时应用作为同一类实时应用进行无区分调度问题,使之能适应更复杂的应用环境。通过证明和实验仿真,改进方案能达到前面阐述的预期研究目标。

应当指出,提出的改进方案中,对于更复杂的调度环境,如不可预测的应用和应用中含有突发任务等情况,还没有作详细探讨,这将是下一步的研究工作。

参考文献:

[1] DENG Z, LIU JW-S. Scheduling real-time applications in an open Environment[C]// RTSS'97: Proceedings of the 18th IEEE Real-Time Systems Symposium. Washington, DC: IEEE Computer Society, 1997: 308 - 319.

[2] WANG YC, LIN KJ. Implementing a general real-time scheduling framework in the RED-Linux real-time kernel[C] // RTSS'99: Proceeding of the 20th IEEE Real-Time Systems Symposium . Washington, D C: IEEE Computer Society, 1999: 246 - 255.

[3] PAREKH A K . A generalized processor sharing approach to flow control in integrated services networks [D]. Cambridge: Massachusetts Institute of Technology, 1992.

[4] KUO T W, YANG W R, LIN K J. EGPS: A class of real-time scheduling algorithms based on processor sharing[C]// Proceedings of the 10th Euromicro Workshop on Real Time Systems. Los Alamitos, CA: IEEE Computer Society Press, 1998: 27 - 34.

[5] ABENI L, BUTTAZZO G. Integrating multimedia applications in hard real-time systems[C]// RTSS'98: Proceedings of the 19th IEEE Real-Time Systems Symposium. Washington, D C: IEEE Computer Society, 1998: 4 - 13.

[6] ABENI L, BUTTAZZO G. Resource reservation in dynamic real-time systems[J]. Real-Time Systems, 2004, 27(2): 123 - 167.

[7] LIPARI G, BARUAH S. A hierarchical extension to the constant bandwidth server framework[C]// Proceedings of the 7th IEEE Real-Time Technology and Applications Symposium. Washington, DC: IEEE Computer Society, 2001: 26 - 35.

[8] MARCHAND A, SILLY-CHETTO M. Dynamic real-time scheduling of firm periodic tasks with hard and soft aperiodic tasks[J]. Real-Time Systems, 2006, 32(1/2): 21 - 47.

[9] LI WEN-MING, KAVI K, AKL R. A non-preemptive scheduling algorithm for soft real-time systems[J]. Computers and Electrical Engineering, 2007, 33(1): 12 - 29.

[10] DEVI U C, ANDERSON J H. Tardiness bounds under global EDF scheduling on a multiprocessor[J]. Real-Time System, 2008, 38(2): 133 - 189.

[11] LIPAI G, CARPENTER J, BARUAH S. A framework for achieving inter-application isolation in multiprogrammed, hard real-time environments[C]// RTSS'00: Proceedings of the 21 st IEEE Real-Time Systems Symposium. Washington, DC: IEEE Computer Society, 2000: 217 - 226.

[12] 龚育昌, 王立刚, 陈香兰, 等. 一种严格按比例派发服务的混合实时调度算法[J]. 软件学报, 2006, 17(3): 611 - 619.

[13] 淮晓永, 邹勇, 李明树. 一种开放混合实时系统的开放自适应调度算法[J]. 软件学报, 2004, 15(4): 487 - 496.

[14] 谭朋柳, 金海, 张明虎. 用于开放式系统的二维优先级实时调度[J]. 电子学报, 2006, 34(10): 1773 - 1777.

[15] 邹勇, 李明树, 王青. 开放式实时系统的调度理论与方法分析[J]. 软件学报, 2003, 14(1): 83 - 90.

[16] 金永贤, 黄经州, 王建国. 基于双层调度框架的开放式实时系统非实时应用调度[J]. 计算机应用, 2008, 28(6): 1608 - 1611.

[17] LIU C L, LAYLAND J W. Scheduling algorithms for multiprogramming in a hard real-time environment[J]. Journal of the ACM, 1973, 20(1): 46 - 61.

(上接第 1501 页)

号,将超高斯信号和亚高斯信号进行了分离,没有考虑更复杂的混合信号,得出的结论尚难以解释或解决更复杂的问题。对不规则波形多组盲源信号的分离有待进一步研究。

参考文献:

[1] HERAULT J, JUTTEN C. Blind separation of sources — Part I: An adaptive algorithm based on neuromimetic architecture [J]. Signal Processing, 1991, 24(1): 1 - 10.

[2] COMON P. Independent component analysis, a new concept [J]. Signal Processing, 1994, 36(3): 287 - 314.

[3] BELL A J, SEJNOWSKI T J. An information-maximization approach to blind separation and blind deconvolution [J]. Neural Computation, 1995, 7(6): 1129 - 1159.

[4] AMARI S, CICHOCKI A, YANG H H. A new learning algorithm for blind signal separation [EB/OL]. [2008 - 10 - 10]. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.40.1433.

[5] HYVARINEN A, OJA E. A fast fixed point algorithm for independent component analysis[J]. Neural Computation, 1997, 9(7): 1483 - 1492.

[6] 朱孝龙, 张贤达, 冶继民. 基于自然梯度的递归最小二乘盲信号分离[J]. 中国科学: E 辑, 2003, 33(8): 741 - 748.

[7] 郑丕谔, 马艳华. RBF 神经网络的递阶遗传训练新方法[J]. 控制与决策, 2000, 2(3): 165 - 168.

[8] 冶继民, 张贤达, 朱孝龙. 信源数目未知和动态变化时的盲信号分离[J]. 中国科学: F 辑, 2005, 35(12): 1277 - 1287.

[9] 张贤达, 朱孝龙, 保铮. 基于分阶段学习的盲信号分离[J]. 中国科学: E 辑, 2002, 32(5): 693 - 703.