用于开放式系统的二维优先级实时调度

谭朋柳, 金 海, 张明虎
(华中科技大学计算机科学与技术学院, 湖北武汉 430074)

摘 要: 提出了一种新的用于开放式系统的调度机制, 即二维优先级实时调度, 它不仅划分任务优先级, 还划分调度策略优先级. 任务的执行顺序由其调度策略优先级和任务优先级共同决定. 它不仅可以解决传统优先级调度机制中机制与调度策略不能相分离的问题, 还提高了效率. 这种机制中引入的 CPU 带宽控制策略, 可以根据需要实现硬实时、软实时、混合实时不同目标的实时系统, 并简化了任务可调度性分析, 且可以为不同权限或级别的用户提供不同 QoS 服务. 这种调度架构不仅效率高, 而且具有很强的开放性, 适用广、易扩展.

关键词: 开放式系统; 二维优先级; 实时调度
中图分类号: TP316.2 文献标识码: A 文章编号: 0372-2112 (2006) 10-1773-05

Two-Dimensional Priority Real-Time Scheduling for Open Systems

TAN Peng-liu, JIN Hai, ZHANG Ming-hu
(*College of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, Hubei 430074, China*)

Abstract: A novel scheduling scheme, called Two-Dimensional Priority Real-Time Scheduling (TDPRTS), is proposed for open systems. This scheme not only sets task priority, but also sets scheduling policy priority. The execution order of task is determined by both task priority and its scheduling policy priority. It can not only separate scheduling mechanism from scheduling policy but also improves the performance. This scheme also introduces the CPU bandwidth control so as to realize different real time systems with different goals, such as hard, soft and hybrid real time systems. It also simplifies the schedulability analysis of the tasks, and it can provide the services with different QoS to the users with different rights and levels. This mechanism has high efficiency and high open degree. It can be used to extensive fields and has high extensibility.

Key words: open systems; two-dimensional priority; real-time scheduling

1 引言

实时调度是实时系统中的一个重要部分, 是保证实时任务时限要求的关键, 也是一个被广泛研究的课题. 过去研究中, 针对各种任务类型提出了很多实时调度方法. 但随着计算机技术的不断发展, 计算机应用的不断深入, 各种任务类型的硬实时、软实时以及非实时任务共存的情况开始出现. 实时已经走向 Web, 实时作为一种服务连同其他的非实时服务一起共存于更庞大的 Grid 系统中也成为可能. 在这样较为开放的系统中, 旧的针对特定任务类型、特定应用领域的实时调度理论和方法不再适用, 必须研究出新的调度理论和方法.

开发式实时系统的特点:

(1) 系统应该具有较强通用性, 不是针对某一特定应用.

(2) 系统应该具有较强的可重配性, 针对不同的应用及 QoS 需求进行相应的配置.

(3) 系统应该具有较强的扩展性, 便于升级和增加新的功能, 以满足新的需求.

(4) 系统能接受动态提交的任务.

针对开放式实时系统的特点, 其调度器相应具有如下特征:

(1) 调度机制与策略相分离, 机制在内核中, 策略由用户选择, 用户可以根据需要为其应用程序随意独立地选择系统中实现了的调度策略.

(2) 调度器具有可重配性, 以满足不同的任务及 QoS 需求.

(3) 调度器应该具有可扩展性, 容易增加新的调度算法, 适应新的任务模型需求.

(4) 能调度动态提交的任务.

实时调度算法分为三类: PD (Priority-Driven)、SD (Share-Driven) 和 TD (Time-Driven). PD 调度是最常用的一种调度方法. PD 调度可以分为可抢占的与不可抢占调度. PD 调度还分为固定优先级调度与动态优先级调度. 最著名的固定优先级调度算法是 RM[1] 算法, RM 算法根据任务的周期确定任务的优先级, 周期越小优先级越高. 最流行的动态优先级调度算法

收稿日期: 2005-06-27; 修回日期: 2006-07-13
基金项目: 国家自然科学基金 (No. 60273076); 国家 863 高技术研究发展计划 (No. 2004AA104280)

是 EDF[1]算法,EDF 算法根据任务的 deadline 确定任务的优先级,deadline 越早优先级越高. PD 调度是种很有效的硬实时调度方法,但并不适合所有的实时应用. 像视频会议等多媒体软实时应用系统中,PD 调度方法显然不合适,SD 调度方法更适合这样的系统. SD 调度是一种按比例共享资源的调度方法,它是基于 GPS[2]算法. 使用 SD 调度的系统中,每个实时任务请求一定的 CPU 资源,系统中的所有任务根据各自所申请的 CPU 资源比例共享计算资源. 著名的 SD 调度算法有 WFQ[3],PGPS[2],WF2Q[4],Fair Service Curves[5]等等. 类似思想的算法还有 TBS[6],CBS[7]以及 CUS[8]. 对具有稳定和预知输入数据流的系统,TD 调度器比较合适,它可以为每个数据流提供可预测的处理能力[9,10]. 在 TD 调度方案中,每个任务的启动、抢占、重新开始以及结束时间都是事先计算好的,通过调度器提供强制保证,因此很容易预测和控制任务的行为. 在小型嵌入式系统、自动化过程控制系统以及传感器等系统可以采用 TD 调度有效地实现. 这三类调度方法分别适用不同的应用领域,分别针对特定的任务模型,均不能直接用于开放式实时系统.

Z. Deng 等[8]提出了一种用于开放式实时系统的两层调度架构,可以同时调度实时程序和非实时程序. Z. Deng 等[11]扩展了文献[8]中的两层调度架构,适用更广的实时应用. Z. Deng 等所提出的两层调度架构中,所有的非实时程序由一个常量利用率虚拟服务器 S_0 调度,采用 SD 调度方法. 每一个实时程序由一个独立的虚拟服务器调度,采用 RM 或 EDF 或其他 PD 调度方法. 而下层是 OS 调度器,采用 EDF 调度算法,负责分配处理器时间给上层服务器,设置各服务器的 Deadline,并根据 Deadline 调度各服务器. 这种两层调度架构可让用户根据应用程序的需要选择调度算法,并同时可调度硬实时和非实时应用程序,但没有考虑软实时应用程序. 这种架构看起来各实时应用程序可独立于其他应用程序单独进行可调度性验证,其实它们之间存在内在联系,即:各实时应用程序的利用率之和要小于或等于 1. 另外这种调度架构存在一个很大的缺陷,每个实时应用程序由一个服务器调度,除非是传统的顺序程序,对大型现代并行分布式程序来讲,显然不合适,所以用这种调度架构实现的系统不是一个真正的开放式实时系统,不适用调度并行分布式系统中具有并行进程的程序.

Tei-Wei Kuo 等[12]把 Z. Deng 等提出的两层调度架构扩展到并行分布式系统中,但是这种扩展只是简单的增加处理器数目,并没有改变实质,每个实时应用程序仍然由独立的虚拟服务器调度,因而其进程仍然只能分配到同一处理器上,并行分布式技术根本毫无用武之地. Tei-Wei Kuo 等[13]用 RM 算法代替了 Z. Deng 等两层调度架构中底层 EDF 算法,上层调度没有变化,因此并没有克服 Z. Deng 等双层调度架构的缺点. 淮晓永等[14]扩展了 Z. Deng 的两层调度架构,提出一种开放环境下的自适应实时系统调度架构——OARtS(open adaptive real-time scheduling). 它能适应开放计算环境的不确定性,有控制地接受实时任务运行;可根据系统空闲计算带宽变化,自适应地调节任务的实时等级,使得系统运行在最优的实时性能上;对于软实时任务,可根据其计算带宽需求变化,自适应地调节其计算带宽分配,以适应任务执行时间时变引起的实时不确定性. 但是 OARtS 中同类约束的实时应用程序采用同一调度服务器调度,也不适用于并行分布式实时应用程序.

Yur-Chung Wang 等[15]提出了一种通用的实时调度架构,这种调度架构包含了 PD、SD 和 TD 三种调度方法,并且可以方便扩展增加新的调度方法. 但是这种调度架构只是将各种调度方法进行简单的集成,系统实际运行时只能选择其中一种调度方法. 如果有两个实时应用程序,一个想采用 PD 方法调度,另一个想采用 SD 方法调度,这时系统就只能满足某一实时应用程序的要求. 这种调度架构不能让用户随意选择调度方法,调度机制与策略不能分离,因此也不适用于开放式实时系统.

采用 Z. Deng 的两层调度架构以及其他基于这种架构的调度机制实现的开放式实时系统,比较适合传统顺序程序,但存在一个很大的缺陷,不适用现代大型并行分布式实时应用;另外因为采用两层调度机制,系统开销比较大. Yur-Chung Wang 的通用实时调度架构虽然集成了各种调度方法,但不能让用户随意选择,只能由系统决定,根本不适用开放式系统,找到一种更有效且适用更广的开放式实时调度架构是本文主要关心的问题.

本文提出了一种有效的用于开放式系统的二维优先级实时调度机制,它将多种调度方法有机地集合起来,能同时调度多种任务类型,而且可以通过重配置,修改各调度方法的 CPU 带宽限制,达到不同的 QoS 级别要求.

2 传统优先级实时调度机制

在引入二维优先级实时调度机制之前,先分析一下传统优先级调度机制的特点. 在过去 PD 算法中,优先级几乎都是一维的传统优先级调度. 一维优先级调度机制中,每个实时任务只有一个优先级,所有任务的优先级均在一维线性空间. 这样的系统中,基于优先级不同的调度算法是不能同时存在的,如 EDF 和 RM 算法,系统运行时只能选择其中的一种算法. 在调度优先级相同的任务时也是一样,系统要么采用 FIFO 先进先出调度算法,要么采用 RR 循环轮转调度算法,FIFO 与 RR 算法也不能同时存在. 至于 SD 和 TD 调度方法,在系统运行时,更不能和 PD 调度方法一起被系统同时采用. 显然,传统优先级调度机制中,调度策略由系统确定,用户不能根据自己需要随意选择调度方法,违背了开放式实时系统中调度机制与策略相分离的思想,不能满足各种任务模型的需求,不适用于开放式实时系统. 传统优先级调度还有一个缺点,开销大,系统只有一个就绪队列,当系统中任务比较多、就绪队列比较长时,任务的出队和入队要花费比较长的查找时间,而且当系统采用指定优先级调度策略(优先级由用户指定)与 RM 结合的混合调度机制时,必须为每个采用 RM 调度策略的任务分配一个优先数,而且当动态来了一个采用 RM 策略的任务时要为系统中所有采用 RM 调度算法的任务重新分配优先数,这也会占用很大的开销.

RTAI[16]是一个开源的实时系统项目. RTAI 中实现了 5 种调度策略:Specified-priority,FIFO,RR,RM,EDF,但他们不能供用户随意选择,其中 FIFO 与 RR 不能同时存在于系统,同一

优先级的任务要么用 FIFO 策略, 要么采用 RR 策略, 也违背了开放式实时系统中调度机制与策略分离的思想, 原因在于 RTAI 采用的也是传统优先级调度. 不过 RTAI 对普通的基于优先级的调度机制做了一点改进, 系统运行时, RTAI 允许 EDF 与 RM 算法同时存在. 它规定采用 EDF 算法的任务的优先级高于采用其他任何调度策略的任务的优先级, 就绪队列中只要有采用 EDF 策略的任务, 系统会优先处理这些任务. 我们从这点改进中得到启发, 提出了二维优先级调度机制.

3　二维优先级实时调度机制

3.1　体系结构

鉴于传统优先级实时调度机制不适用于开放式实时系统, 而且系统开销比较大, 为了克服这些缺点, 我们提出了灵活的二维优先级实时调度机制, 它不仅可以解决调度策略与机制分离的问题, 理论上还可以大大提高系统的性能. 二维优先级实时调度机制的思想是: 系统采用两级优先级, 先将各种调度策略划分优先级, 如: 将 EDF 策略优先级设为最高, 其次是 RM, 再其次是 Specified priority, 然后是 FIFO, 最后是 SD, 这是优先级的一级划分, 然后划分各调度策略内部任务的优先级, 这是二级优先级. 传统优先级调度机制中任务的执行顺序仅由任务的优先级决定, 而二维优先级调度机制中任务的执行顺序由任务的调度策略优先级与任务的优先级共同决定. 但是, 当最高优先级就绪队列始终有任务时, 低优先级的任务永远得不到运行, 为了解决这个问题, 我们在二维优先级调度机制中引入了 CPU 带宽控制. 其思想是引入 SD 共享驱动的调度方法, 将 CPU 带宽在各种实时调度策略之间进行分配, 设置各种调度策略 CPU 带宽上限, 各策略带宽上限之和小于或等于 1. 这样, 系统首先调度最高优先级策略中具有最高优先级的任务, 当分配给这种策略的 CPU 带宽用完时, 不管这种策略就绪队列中是否还有任务, CPU 使用权都会让给次优先级策略中的任务. 因而系统不仅可以保证具有高优先级调度策略的任务优先被执行, 还保证具有较低优先级调度策略的任务也能得到运行. 二维优先级实时调度机制体系结构如图 1.

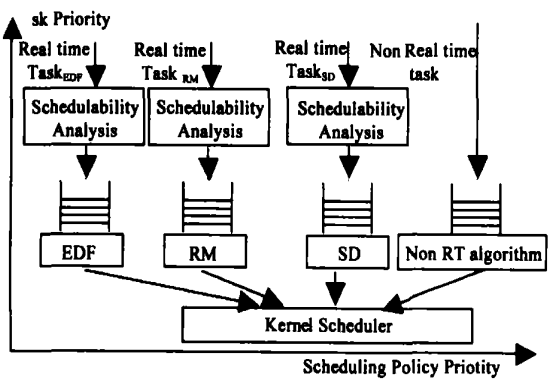

图 1　二维优先级实时调度机制体系结构

采用二维优先级实时调度机制的系统运行时, 用户可以根据需求随意选择合适的调度策略, 比如紧急的控制任务可以选择 EDF 算法调度, 周期性的硬实时任务可以采用 RM 算法, 而多媒体软实时应用程序则可以采用 RR 算法, 解决了调度机制与策略分离的问题. 另外这种实时调度机制为每种调度策略维护一个就绪队列, 任务的出队/入队时只需浏览相应调度策略的就绪队列, 开销很小, 而且并不需要为采用 EDF 算法和 RM 算法的任务分配优先数, 可以节省这部分开销, 当有一个新的 RM 或 EDF 任务到来时, 也免掉了重新分配所有任务优先级的开销. 另外, 调度器扩展性强, 可以自由加入新的调度策略, 如最少空余时间优先策略, 当两个采用 EDF 策略的任务的 Deadline 相同或两个采用 RM 策略的任务的周期相同时就可以根据最少空余时间优先策略确定它们的执行顺序. 通过引入带宽控制, 二维优先级实时调度机制简化了任务可调度性分析过程, 当一个新任务被提交时, 它的可调度性分析只与跟它采用同一调度策略的任务有关, 跟其它所有任务无关, 提高了系统效率; 调度器很灵活, 具有可重配性. 调整各调度策略 CPU 带宽, 可配置出适合不同应用的系统. 提高 EDF 与 RM 调度策略 CPU 带宽, 系统会偏向于硬实时系统, 系统能提供更强的硬实时服务, 降低它们的 CPU 带宽, 系统偏向于软实时系统, 系统能为软实时任务提供更高的 QoS. 当 EDF 与 RM 调度策略 CPU 带宽分配适当, 系统既能提供一定的硬实时能力, 又可以提供较高 QoS 的软实时服务; 另外, 有很强商业价值, 可以根据用户不同的权限或级别提供不同 QoS 的服务, 就像宽带网络一样, 不同用户得到不同带宽, 收费也不一样.

3.2　可调度性分析

实时系统中可调度性分析, 也就是新任务的接受控制, 其目的是为了确保接受一个新任务时, 系统中原先所接受的所有实时任务仍然是可调度的.

定义 **1:** CPU 利用率 U 是指某时间段 T 内, CPU 的计算时间 C 与 T 的比值, 即:

$$U = C/T \tag{1}$$

显然, U 总是小于或等于 1.

定义 **2:** 实时任务 t_i 的 CPU 利用率 u_i 是指实时任务 $t_i(c_i, P_i)$ (其中 c_i 为任务 t_i 最坏情况下的计算时间, 如果 t_i 是周期性任务, 则 P_i 为它的周期, 否则为相对截止期限) 的 c_i 与 P_i 的比值, 即:

$$u_i = c_i/P_i \tag{2}$$

假设 **1:** 任何任务因访问临界资源而最多被阻塞一次, 临界区最长为 B (即最长阻塞时间为 B).

对于 EDF 策略, 假设 EDF 策略 CPU 带宽上限为 U^{EDF}, 当前所有采用 EDF 策略任务的总 CPU 利用率为 U_{EDF}, 当一采用 EDF 策略新任务 $t_i(c_i, P_i)$ 到来时, 它是可接受的, 根据文献 [13] 中的理论 4, 当且仅当:

$$(B + c_i)/P_i + U_{EDF} \leqslant U^{EDF} \tag{3}$$

对于 RM 策略, 假设 RM 策略 CPU 带宽上限为 U^{RM}, 当前所有采用 RM 策略 m 个任务的总 CPU 利用率为 U_{RM}, 当一采用 RM 策略新任务 $t_i(c_i, P_i)$ 到来时, 它是可接受的, 根据文献 [13] 中的理论 3, 当且仅当:

$$(B + c_i)/P_i + U_{RM} \leqslant (m + 1)(2^{1/(m+1)} - 1)\, U^{RM} \tag{4}$$

对于 SD 策略, 假设 SD 策略 CPU 带宽上限为 U^{SD}, 当前所有采用 SD 策略任务的总 CPU 利用率为 U_{SD}, 当一采用 SD 策略且 CPU 利用率为 u 的新任务到来时, 它是可接受的并保证其 QoS, 如果它的CPU 利用率小于或等于能分给它的带宽, 即:

$$u \leqslant \frac{u}{u + U_{SD}} U^{SD} \qquad (5)$$

3.3 调度实例

这里有 5 个实时任务 $t_1(2, 10)$, $t_2(1, 20)$, $t_3(5, 20)$, $t_4(2, 40)$和 $t_5(1, 10)$同时到达系统, 假设之前系统无任何实时任务. 其中, t_1 和 t_2 是采用 EDF 策略的非周期性硬实时任务, t_3 和 t_4 采用 RM 策略的周期性硬实时任务, t_5 是采用 SD 调度策略的软实时周期性任务. 根据定义 1, 这 5 个任务的 CPU 利用率分别为 0.2, 0.05, 0.25, 0.05 和 0.1. 假定 EDF, RM 和 SD 策略 CPU 利用率上限分别为 0.2, 0.6 和 0.2. 图 2 显示了二维优先级调度机制对这 5 个任务调度的结果. 其中 t_2 没有通过可调度性分析被系统拒绝.

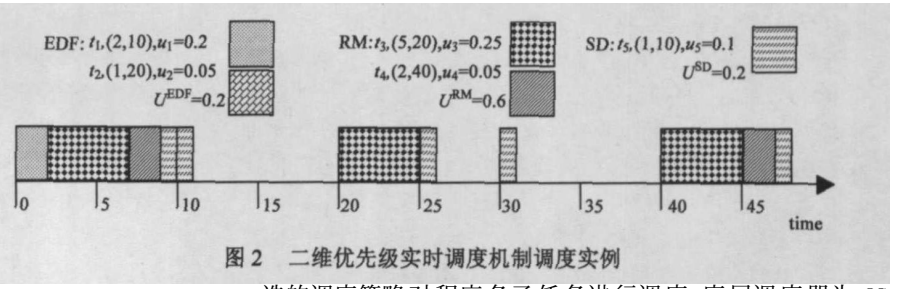

图 2 二维优先级实时调度机制调度实例

4 性能分析

开放式实时系统调度概念于 1997 年首先被 Z Deng 在文献[8] 文中提出, 后来有人在 Z Deng 的研究基础上做了些扩展, 也有了一些研究成果. 但由于起步晚, 还没有具体的评价标准, 因此本文只从效率和开放程度两方面对 Z Deng 两层调度架构[8]、Wang Y 提出通用调度架构[15] 以及淮晓永提出的 OARtS[14] 与二维优先级实时调度机制进行定性的比较.

效率主要由调度的复杂程度决定. Z Deng 两层调度架构分两级调度, 上层调度器为服务器调度器. 每个实时应用程序被提交时, 系统为它建立一个独立的服务器, 按照实时程序所选的调度策略对程序各子任务进行调度. 底层调度器为 OS 调度器, 采用 EDF 调度算法, 负责分配处理器时间给上层服务器, 设置各服务器的 Deadline, 并根据 Deadline 调度各服务器. 系统多了许多中间服务器要管理, 开销比较大, 效率不高. Wang Y 的通用调度架构将任务的调度分成两个过程, 分别由调度分配器(Schedule Allocator) 和(Schedule Dispatcher) 完成. 分配器负责注册新实时作业并设置新作业的调度属性, 在用户空间完成. 而派遣器根据系统调度策略, 负责实际的实时作业调度, 在内核中完成. 由于整个调度分两个过程完成, 且第一个过程在用户空间完成, 所以效率也不高. OARtS 是在 Z Deng 两层架构基础上的扩展, 并加入了自适用调节机制, 所以开销也比较大. 二维优先级实时调度架构中, 新任务到达时, 只需按照所选调度策略入相应队列, 即使不同种类的任务来得很分散, 也不存在各调度策略的频繁切换, 并且加快查找速度, 提高效率. 而调度时机到来时, 系统总是选择高优先级调度策略就绪队列中队首的任务执行, 整个调度过程基本没有其它的额外计算, 简单高效.

调度架构的开放程度决定于所适用任务模型的多少、调度机制与策略是否分离、可供选择的调度策略多少、是否具有 QoS 控制、扩展性强弱以及是否适用动态环境等. Z Deng 的两层调度架构、Wang Y 的通用调度架构、OARtS 及二维优先级实时调度机制开放程度的比较如表 1.

表 1 两层调度架构、通用调度架构、OARtS 和二维优先级实时调度机制之间开放程度的比较

调度架构	适用任务模型	并行分布式应用	机制与策略分离	调度策略	QoS 控制	扩展性	动态环境
Z Deng 的两层调度架构	硬实时/非实时	不适用	支持	PD/SD/RR(Cyclic)	没有	较强	适用
Wang Y 的通用调度架构	硬实时/软实时/非实时	适用	不支持	PD(EDF, RM)/SD/TD	没有	较强	适用
OARtS	硬实时/软实时/非实时	不适用	支持	PD/SD	有	较强	适用
二维优先级实时调度机制	硬实时/软实时/非实时	适用	支持	PD(EDF, RM)/SD 等	有	较强	适用

5 结束语

本文提出的二维优先级实时调度机制, 高效率, 高开放度. 这种机制中任务的执行顺序不仅由任务优先级决定, 还由调度策略优先级决定. 系统优先执行调度策略优先级高的就绪队列中的任务, 而采用同一调度策略的任务按照其调度策略确定的优先顺序进行执行. 这种机制不仅解决了传统优先级机制中调度策略依赖于调度机制的问题, 还提高了效率. 二维优先级机制通过引入灵活可变的带宽控制, 可以实现硬实时、软实时、混合实时不同目标的系统, 还简化了任务可调度性分析, 且可以为不同权限或级别的用户提供不同 QoS 服务. 这种调度架构还有很强的扩展性, 可以很容易加入新的调度策略, 比如: 通过处理器带宽预留技术, 可以引入 TD 时间驱动的调度策略. 二维优先级实时调度机制不仅可以用于单机系统, 也适用于分布式实时系统, 对基于 Web 或 Grid 的真正的开放式系统的大型并行分布式应用, 也非常适用. 下一步我们将这种机制实现在基于 Grid 的实时信息处理平台.

参考文献:

[1] C L Liu and J Layland. Scheduling algorithms for multiprogramming in a hard real-time environment[J]. Journal of the ACM, 1973, 20(1): 46–61.

[2] A K Parekh and R G Gallager. A generalized processor sharing approach to flow control in integrated services networks: the single-node case[J]. IEEE/ACM Trans. Networking, 1993, 1(3): 344–357.

[3] A Demers, S Keshav, and S Shenker. Analysis and simulation of a fair queueing algorithm[J]. Journal of Internetworking Research and Experience, 1990, 1(1): 3–26.

[4] J C R Bennett and H Zhang. WF2Q: Worst-case fair weighted fair queueing[A]. Proc. of INFOCOMM'96[C]. San Francisco, CA: IEEE Computer Society, 1996: 120–128.

[5] I Stoica, H Zhang and T S E Ng. A hierarchical fair Service curve algorithm for link-sharing, real-time and priority services [J]. ACM SIGCOMM Computer Communication Review, 1997, 27(4): 249–262.

[6] M Spuri and G C Buttazzo. Efficient aperiodic service under the earliest deadline scheduling[A]. Proc. of IEEE Real-time Systems Symposium[C]. San Juan, Puerto Rico: IEEE Computer Society, 1994: 2–11.

[7] L Abeni and G C Buttazzo. Integrating multimedia applications in hard real-time systems[A]. Proc. of IEEE Real-Time Systems Symposium[C]. Madrid, Spain: IEEE Computer Society, 1998: 4–13.

[8] Z Deng, J W-S Liu, J Sun. A scheme for scheduling hard real-time applications in open system environment[A]. Proc. of 9th Euromicro Workshop on Real-Time Systems[C]. Toledo, Spain: IEEE Computer Society, 1997: 191–199.

[9] H Kopetz. The time-triggered model of computation[A]. Proc. of IEEE Real-Time Systems Symposium[C]. Madrid, Spain: IEEE Computer Society, 1998: 168–177.

[10] Ching-Chih Han, et al. Distance-constrained scheduling and its applications to real-time systems[J]. IEEE Trans. Computers, 1996, 45(7): 814–826.

[11] Z Deng et al. Scheduling real-Time applications in open system environment[A]. Proc. of IEEE Real-Time Systems Symposium[C]. San Francisco, CA: IEEE Computer Society Press, 1997: 308–319.

[12] Tei-Wei Kuo, et al. An open real-time environment for parallel and distributed systems[A]. Proc. of 20th Inter. Conf. on Distributed Computing Systems[C]. Taipei, Taiwan: IEEE Computer Society, 2000: 206–213.

[13] Tei-Wei Kuo, Ching-Hui Li. A fixed-priority-driven open environment for real-time applications[A]. Proc. of the 20th IEEE Real-Time Systems Symposium[C]. Phoenix, AZ, IEEE Computer Society, 1999: 256–267.

[14] 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.

[15] Wang Y-C, et al. Implementing a general real-time scheduling framework in the RED-Linux real-time kernel[A]. Proc. of the 20th IEEE Real-Time Systems Symposium[C]. Phoenix, AZ, IEEE Computer Society, 1999: 246–255.

[16] http://www.rtai.org/.

作者简介:

谭朋柳　男, 1975 年生于湖北, 博士研究生, 主要研究领域为分布式实时系统、实时调度、集群与网格计算、可信系统等.
E-mail: pltan@hust.edu.cn.

金　海　男, 1966 年生于西安, 博士, 教授, 博士生导师, 现任华中科技大学计算机学院院长, IEEE 及 IEEE 计算机学会会员, ACM 会员, 国家杰出青年基金获得者. 主要研究领域为计算机体系结构、集群和网格计算、并行与分布式处理、对等计算、网络存储、网络安全等.