开放式实时系统的调度理论与方法分析*

邹 勇+, 李明树, 王 青

(中国科学院 软件研究所 互联网软件技术实验室,北京 100080)

Analysis for Scheduling Theory and Approach of Open Real-Time System

ZOU Yong+, LI Ming-Shu, WANG Qing

(Laboratory for Internet Software Technologies, Institute of Software, The Chinese Academy of Sciences, Beijing 100080, China)

+ Corresponding author: Phn: 86-10-82620803 ext 811, E-mail: zouy@intec.iscas.ac.cn

http://itechs.iscas.ac.cn

Received 2002-04-22; Accepted 2002-08-22

Zou Y, Li MS, Wang Q. Analysis for scheduling theory and approach of open real-time system. *Journal of Software*, 2003,14(1):83~90.

Abstract: As the applications of real-time systems are becoming more and more popular, the system scheduling mechanism faces new requirements and challenges because of the coexistence of different kinds of real-time and non-real-time tasks. The open real-time system has been paid attention because of giving a way to solve the problem and also brings new ideas to scheduling theory and approach. Firstly, the basic concepts and theories of open real-time system are described. And then, a hiberarchy model for scheduling objects is presented. It adapts to the scheduling environment of open real-time system. Several related scheduling approaches are compared in detail, and their functional features, the applying scope, and some of their common characters are pointed out. The result come out that it is feasible to integrate different kinds of approaches, and this would be a direction. In the integrated scheduling mechanism, the advantages of each approach can be kept, therefore, the application requirements of open real-time system can be met better.

Key words: real-time; open system; scheduling algorithm; server; deadline

摘 要: 随着实时系统应用的日益广泛,多类型实时和非实时任务并存的情况给系统调度机制带来了新的需求和挑战.开放式实时系统的相关研究因为解决这一问题而受到关注,也为调度理论与方法带来了新的思路.在阐述了开放式实时系统的基本概念和理论的基础上,提出了一种调度对象的层次性模型,适用于开放式实时调度环境.对几种有代表性的相关方法进行了详细的比较研究,指出它们各自的功能特征和适用范围及其共同特征.把不同方法进行有机集成是可行的,也是一个发展方向,可以使各方法在集成的调度机制内各施所长,更适合于开放式实时系统的应用环境.

关键词: 实时;开放系统;调度算法;服务器;截止期

* Supported by the National Natural Science Foundation of China under Grant No.69896250 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant No.2001AA113201 (国家高技术研究发展计划)

第一作者简介: 邹勇(1976－),男,山东青岛人,博士生,主要研究领域为实时系统,操作系统.

中图法分类号: TP311 文献标识码: A

在实时计算中,系统的正确性不仅依赖于计算的逻辑结果还依赖于结果产生的时间[1].在实时系统研究中,实时调度是一个难点,也是被广泛研究的题目.调度的实质是资源的分配,进一步地,实时调度强调的是为一系列任务决定何时得到何种资源运行.合理的实时调度可以充分利用有限的系统硬件资源,从而确保实时任务的时间约束得到满足.与分时系统中的调度所追求的公平性原则不同,实时调度根据各个任务的实时需求的不同而区别对待它们[2,3].

近年来,虽然针对不同类型的实时任务提出的调度方法不断涌现,但是随着计算机应用的不断深入和发展,多类型的硬实时、软实时与非实时任务共存于同一系统中的情况越来越广泛,从而产生了新的需求.而现有的那些针对性较强且适用范围比较专一的调度方法不能满足这种需求.

本文所讨论的开放式实时系统的研究为在一个系统中同时支持多种实时调度方法提供了可能性.首先阐述了开放式实时系统的相关概念、基本理论和研究现状;其次,根据其中的调度对象的特征,提出了一种层次性调度对象模型;然后,详细分析了几种有代表性的主要调度方法,重点对它们各自的功能特征和适用范围作了评价和比较;在此基础上,找出它们的若干共同之处,并阐述了这些共同点带来的启发:有机集成不同方法于同一调度框架中的可行性;最后,明确指出了开放式实时调度方法的一个发展方向.

1 开放式实时系统理论及研究现状

我们知道,传统的专用实时系统因为需要满足各任务的时间约束,必须对系统中所有实时应用首先作全局分析,再进行统一调度.而在运行中,其中的某个应用发生错误,将可能影响到其他非相关应用的实时性[4].这一类系统被称为封闭式的.显然,对于封闭式实时系统的开发,应用层的开发人员必须对系统中已有的全部应用程序进行统一的严格实时性验证,随时掌握全局信息.并且,每当有新的实时应用要加入系统时,需要重新进行全局实时性能分析和检验.为了使实时系统具有动态的可扩展能力,支持多类型的实时、非实时混合的复杂实时调度需求,近几年人们提出了开放式实时系统理论.“开放”术语在不同上下文中有多种语义,这里有其特殊的定义.

定义 1. 开放式实时系统,是指系统中非相关的实时或非实时应用可单独进行开发和验证,并且,当系统进行动态扩展时,无须作全局的实时性分析[4].

在这样的系统中:

(1) 多类实时调度策略、非实时调度策略可被集成,以构建多调度策略并存的实时复合任务调度机制;

(2) 开发者可根据实际需要,为各个应用选择不同的调度方法,按照一定的比例,分别请求一定的系统计算资源;

(3) 在不影响系统中原有实时任务集调度的前提下,任务可以动态加载,并进行可加载性测试;

(4) 非相关应用在系统中共同运行时互不干扰;

(5) 各个应用之间的调度由开放式系统自动完成.

目前,与开放式实时系统相关的实时调度研究提出了几种不同的方法.

首先是基于比例共享的调度算法.这类算法虽然不是专为开放式实时系统提出的,但其核心思想可以在其中应用.其关键之处在于,为系统中的每个实时应用程序按一定比例分配处理机或其他共享资源(如网络带宽).其优点是,系统中的每个实时任务可以确保其所要求的处理机资源,而免受其他任务占用超出计划的处理机执行时间的干扰[5,6].

GPS(generalized processor sharing)算法[5]即是有代表性的一种.它首先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.

EGPS(earliest-completion-time GPS)算法[6]将 GPS 的思想运用到实际系统中.它一方面继承了 GPS 的按比例分配处理机带宽的思路,另一方面不需要将实时应用理想化为粒度可无限细分的工作流,所以,可以真正地在

实际系统中得到应用.所采用的方法是: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.

上述方法没有考虑在同一系统中兼容多种调度算法的问题.

近期提出的几种主要的开放式实时系统相关调度方法都采用了基于服务器的策略.这里的服务器是指系统调度机制创建的特殊任务,它为调度对象提供服务.

文献[4,7]分析了开放式实时系统的特征,并提出了一种基于 CUS(constant utilization server)和 TBS(total bandwidth server)服务器的调度方案.它允许硬实时、软实时和非实时任务共存于一个系统,提供了应用程序之间的资源隔离支持,可从系统资源利用的角度避免非相关应用程序在运行时的互相干扰.它需要已知精确的实时任务运行参数.文献[8]提出了 PShED(processor sharing with earliest deadline first)算法,它允许系统中同时运行的实时应用使用不同的调度方法,并且它的服务器对就绪作业的执行并不局限于 FCFS(first-come first-served)顺序,只需已知其截止期参数,进而再计算服务器的执行预算.但它只能适合于完全可抢占类型的任务.文献[9]致力于软实时任务的调度,提出了可以确定的处理机带宽调度任务的 CBS(constant bandwidth server)方法,每个 CBS 在不超出其所声明的处理机带宽的情况下,按照 FCFS 顺序对关联于它的任务所产生的作业服务.文献[10]则扩展了 CBS 方法,它提出的 H-CBS(hierarchical CBS)算法可以进一步支持软实时任务集间的带宽隔离.

以上研究针对不同的调度对象,并侧重于不同方面,探讨了在多种类任务或任务集共存的系统中,如何以满足实时需求为首要目的进行实时调度的方法.每种方法各有其适用的任务类型和调度场合,同时也有一些共同特征.

2 调度对象的模型定义及相关术语

2.1 层次性调度对象模型

开放式实时系统的调度对象不再是单一的,而是具有明显的层次性.下面进一步为本文定义所涉及的相关概念,并提出调度对象的层次模型.

定义 2. 任务是指完成某一特定功能的软件实体,它是实时调度中的基本单位.任务在其生命期中的某一次执行称为该任务的一个作业.

定义 3. 在实际中,一个完整的应用往往是一组任务的集合,我们把组成一个应用的任务的集合称为任务组.当任务组中只有一个任务时,任务组等同于任务.因此任务组是任务概念的超集.

从应用开发者的角度来看,一个特定的实时应用的任务组有其适合的调度方法与之对应,允许系统中的各个任务组使用不同的调度器将极大地方便应用开发,并取得较好的应用效果.开放式实时调度环境的主要目的之一也是要为每个实时应用可以自由地使用其适合的调度器提供支持.为此,我们给出以下定义:

定义 4. 把应用开发者确定的一个任务组及其相应的调度器称为一个任务组-调度器应用对(task-set and scheduler,简称 TSS).把没有配置调度器的任务组或任务看作 TSS 的一种特殊情况,即认为它们有一个不做任何事的调度器.因此,TSS 的概念是任务组或任务的超集.

综上所述,开放式实时系统的调度对象可分为 3 个层次:任务、任务组和 TSS.它们之间的关系如图 1 所示.

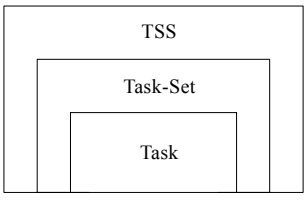

Fig.1 Hiberarchy model for scheduling objects in open real-time systems

图 1 开放式实时系统调度对象的层次性模型

本文讨论的任务类型包括硬实时或软实时的周期任务或非周期任务以及非实时任务.从服务器的角度来

看,调度对象的范围包括单独的任务、任务组和 TSS;从整个系统调度框架的角度来看,调度对象分别是以上 3 种对象的集合.若未作特殊说明,调度对象具有以下特性:(1) 各个调度对象之间互相独立;(2) 调度对象的每个任务对共享资源的访问方式为互斥访问(这里,非互斥访问的资源不使用术语“共享资源”);(3) 非实时应用作为具有非实时的调度器的 TSS 来对待.

2.2 与现有模型的分析比较

现有的开放式实时系统在调度对象模型方面都沿用了传统的实时系统的相应方法[2,4],即仅以任务作为调度算法的处理对象,模型结构单一.各方法之间的不同是,对任务类型附加了不同的限制条件.而从客观的角度来看,开放式实时系统中的调度对象与传统实时系统有很大的不同,特别是局部调度器的引入,使单一的以任务为对象的模型不足以进行完整描述.这里提出的层次性调度模型从任务、任务组和 TSS 这 3 个层次进行描述,与系统的客观情况相符,并覆盖了目前系统中的全部调度对象类型.另外,层次性的模型也有利于进行不同层面的调度方法研究,方便了进一步的开放式实时系统的调度方法研究.

2.3 相关术语

定义 5. 最坏执行时间(worst case execution time,简称 WCET)是指,任务在其生命期中的每个作业的执行时间的最大可能值[9].

WCET 常用来作为实时调度中任务对处理机的可能占用情况的一个衡量值.

定义 6(服务器速率). 不失一般性,假设总的系统处理机的速度为 1,当把服务器 S_i 看作一个虚拟处理机时,服务器速率 σ_i 是指 S_i 的速度与系统处理机的速度比值.

这个比值表示了服务器占用系统处理机总的带宽的比例,所以有 $0<\sigma_i\leq 1$.

定义 7. 服务器能量(server capability)是指,从当前时刻起,服务器可占用系统处理机的时间.

3 基于服务器策略的方法分析

如前所述,目前开放式实时系统的几种主要的相关调度方法是基于服务器策略的,以下对它们分别进行详细分析,然后再综合分析相关性和共同特征.

3.1 基于CUS和TBS的双层调度框架

3.1.1 框架概述

这种方法以两种类型的服务器为基础,构造了双层的调度框架,如图 2 所示.在底层,由采用 EDF 算法的系统调度器对服务器的集合$\{S_1,S_2,\ldots,S_n\}$进行调度,而每个服务器都按一定的比例占用处理机时间,同时它又与一个 TSS 相关联,为其提供资源服务.

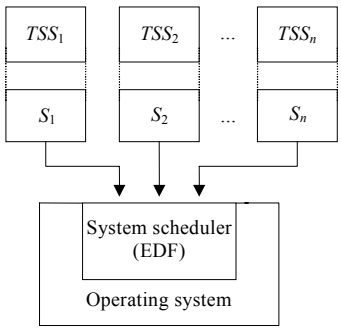

Fig.2 Two-Level scheduling framework

图 2 双层调度框架

在上层,每个 TSS 包含一个局部调度器,它决定 TSS 中的任务组$(\tau_1,\tau_2,\ldots,\tau_n)$的调度顺序.每个 TSS 的局部调度器会产生一个作业的执行队列,这个队列交由相应的服务器来执行,如图 3 所示.所以,从 TSS 角度来看,每个服务器相当于具有一定速率的处理机.

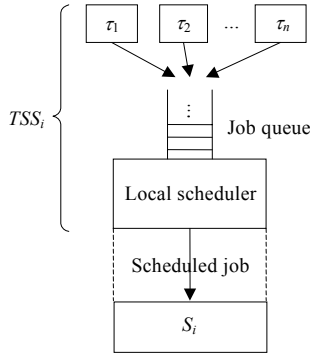

Fig.3 Server and its scheduling object
图 3 服务器及其调度对象

由此可以看出,这个双层调度框架实际上是致力于为系统中的每个非相关应用提供一个具有确定速率的虚拟处理机.这是开放式实时系统调度机制的关键所在.

为了实现整个系统的实时性能可预测和可验证,该框架的系统调度器要求其上的所有服务器的处理机利用率之和小于或等于 1,这实际上也是 EDF 算法的可调度条件.

这种双层的调度框架可使同一系统中容纳多类型的调度方法.然而,还有另外一些关键性的问题需要解决:

(1) 在允许任务的作业互相抢占的情况下实时调度;

(2) 在允许任务有不可抢占区(nonpreemptable section,简称 NPS)和全局共享资源的情况下实时调度.

为了解决问题(1),将 TSS 区分为两类:可预测的和不可预测的.可预测 TSS 是指,若 TSS 在专有处理机上运行,在任意 t 时刻,下一次的上下文切换时间可以已知.对于不可预测的,则相反.对于可预测的 TSS,将它在下一次上下文切换时间之前的作业执行作为一次调度单位,这样便回避了作业抢占带来的问题.对于不可预测的 TSS,仍需已知在当前时刻之后的多少时间长度内不会有抢占发生,然后把当前作业的实际截止期设置在这段时间之内.这需要通过提高服务器速率才能实现.

为了解决问题(2),在一定程度上降低服务器的总处理机利用率的上限,许可任务有短暂的 NPS 执行时间.这是通过牺牲系统硬件资源的利用率换取任务的不可抢占时间.

3.1.2 CUS 与 TBS

图 2 中的 S_i 可能是 CUS 和 TBS 中的任何一个,这取决于不同类型的 TSS 的需求.因为底层的系统调度器是基于 EDF 算法的,所以要求 S_i 必须以截止期参数参加系统的调度.

设 S_i 为系统中的一个 CUS 服务器,在 t 时刻,它的截止期为 $d_{i,k}$,计算如下:

$$d_{i,k}=\max\{t,d_{i,k-1}\}+e_{i,k}/\sigma_{i,k}. \quad (1)$$

其中 $\sigma_{i,k}$ 为 t 时刻 S_i 的服务器速率,$e_{i,k}$ 为 S_i 的就绪队列头部作业的剩余 WCET.S_i 的能量在 $\max\{t,d_{i,k-1}\}$ 时刻补充,补充值等于 $e_{i,k}$.CUS 就是在已知每个任务的 WCET 参数的前提下,通过 $e_{i,k}$ 计算截止期,该截止期同时作为 t 时刻 S_i 的截止期.

TBS 的截止期设定方法与 CUS 相同,不同的是服务器能量的补充时机.设作业 $J_{i,k}$ 在 t 时刻到达,若 $t\geq d_{i,k-1}$,则没有区别;若 $t<d_{i,k-1}$,则 TBS 在 $J_{i,k-1}$ 执行完毕时就可以补充能量,而 CUS 必须等到 $d_{i,k-1}$ 时刻[4,7].

由以上分析我们可以看出,CUS 因能量补充时间被延迟到上一个作业的截止期之后,其当前作业即使在处理机空闲的状态下也不能执行.所以,CUS 适合的是没有必要提前完成、以非抢占方式调度的 TSS 类型.比如,基于任务表的循环调度、时间驱动的调度等.TBS 与 CUS 不同的特点,使它的平均任务响应时间比 CUS 高,适合于调度那些强调较好的平均任务响应时间的实时任务.

CUS 和 TBS 只能以 FCFS 顺序执行作业.另外,必须已知每个任务的 WCET.而在实际中,对 WCET 进行精确的测量往往比较困难.特别是对于某些类型的软实时任务(如多媒体流解码),其作业的执行时间差别很大,即使获得了 WCET,若按照 WCET 分配处理机带宽,则显然是对处理机资源的极大浪费.

3.2 PShED算法

PShED 算法的总体方案类似于第 3.1 节中提到的双层调度架构,主要不同在于所采用的服务器类型.

在任意 t 时刻,PShED 算法中的服务器的截止期在以下 3 种情况下被改变:

(1) 当前作业执行完毕;

(2) 截止期比当前作业的截止期更小的作业到达;

(3) 服务器能量耗尽.

服务器的截止期的计算十分简捷,即不论在何种情况下,始终与其作业集合中的最近截止期保持相等[8].

它使用专门的数据结构跟踪服务器能量的已消耗值,在需要计算服务器能量的补充值的时刻,与服务器的当前截止期和速率相结合,计算当前时刻的服务器应补充的能量.

对于一个 TSS,只要它在速率为 σ 的专有处理机上可调度,那么它也一定可在速率为 σ 的 PShED 服务器上可调度.因此,开发一个与 PShED 服务器相关联的 TSS,只要已知 TSS 所需的服务器速率即可.在调度过程中,不必已知 WCET 参数,也不必区分是可预测的还是不可预测的.

PShED 算法要求它的调度对象必须是完全可抢占的,而不能用于以非抢占方式运行的 TSS.

3.3 基于CBS服务器的算法

每个 CBS 都有自己的参数对(Q,P),其中 Q 为服务器的最大能量补充值,P 为服务器周期.当服务器的能量耗尽时,其截止期将在当前值的基础上后延 P 个时间单位,同时补充服务器能量.每个 CBS 服务器与一个实时任务相关联,它可以确保后者可获得至少 Q/P 的处理机带宽,并且完成时刻不迟于在速率为 Q/P 的专有处理机上运行时的完成时刻加 P[9].

参数对(Q,P)决定了 CBS 的行为特性.其中 P 值的大小实际上决定了调度粒度,其值越小,则平均响应时间越小,但调度引起的上下文切换次数会增加.因此,它的取值是权衡任务的平均响应时间需求和调度开销而得.在 P 值确定的情况下,Q 的取值应由任务所需要的处理机带宽 Q/P 决定.

由以上分析可知,CBS 不需要任务为其截止期和服务器能量的计算提供任何参数,只要确定其(Q,P)参数对即可.在现实世界中,存在这种类型的软实时任务,它们的 WCET 值难以获得,或者所产生的每个作业的 WCET 变化范围大,它们所需求的是可满足其平均 WCET 的处理机带宽,而 CBS 正适用于此类需求.

一个 CBS 只与一个任务相关联,并只按 FCFS 顺序执行该任务产生的作业,因此也不支持任务使用本地调度策略.在硬实时应用方面,它只适于周期和相对截止期都严格等于 P 值,最坏执行时间小于等于 Q 值的周期任务调度.

3.4 H-CBS算法

在有些情况下,一个实时应用由一组任务组成,而 CBS 算法只提供单个任务间的处理机带宽隔离.H-CBS 算法通过记录任务组内的各个任务消耗服务器能量的情况,计算可挪用时间,以达到任务组间的带宽隔离的目的[10].也就是说,在 H-CBS 算法下,每个任务组及组内的每个任务都有自己的可保障的处理机带宽.它是 CBS 的一个扩展.

另一方面,由于需跟踪服务器能量消耗情况,H-CBS 算法比 CBS 更复杂,我们只在任务组内的任务数量大于 1 的情况下应用 H-CBS.

在其他方面,H-CBS 与 CBS 的特点相同.

4 性能评价及各方法的比较研究

在以上对各类开放式实时系统调度方法的分析中,我们可以发现,不同类型的服务器各有其适用范围和所长之处,任何一种方法都不是完全通用的.下面,对它们的性能进行具体的评价与比较.

基于 CUS 和 TBS 的方法着重于处理多类型任务同存情况的调度,适用于硬实时、软实时任务和非实时任务混合的系统,并允许任务有 NPS 和共享资源.但是,对于每个实时任务都必须已知其 WCET 参数.这个条件对

于有些情况,特别是软实时任务来说是比较困难获取的,甚至是不合理的.

PShED 算法只针对于完全可抢占类型的任务,因此,有 NPS 的任务自然不在它的适用范围内.它的优势在于,对就绪作业的执行不只局限于 FCFS 顺序,并且只需作业提供其截止期参数,进而再计算服务器能量.允许系统中同时运行的实时应用使用不同的调度方法,适用范围是硬实时调度.

使用 CBS 服务器的方法主要针对软实时和非实时的应用,在硬实时方面,只能是具有严格限制条件的周期任务可以适用.它的调度对象局限于任务.CBS 在软实时任务调度方面具有明显优势:只需声明服务器的周期和每周期内的能量预算,而不需要提供 WCET 或截止期参数.它在 WCET 变化范围大(如 MPEG2 解码),或者截止期不明确的实时应用方面尤其适合.

H-CBS 算法通过扩展基于 CBS 的方法,以算法复杂度的代价换取了对任务组调度的支持.除了这个区别以外,其他的适用范围和优势与 CBS 一致.

表 1 对各个类型的服务器进行了比较.

Table 1 Comparison of the functions of different kinds of servers

表 1 各类型服务器的功能比较

Function / Type	Hard real-time	Soft real-time	NPS	WCET	TSS	Task-Set
CUS,TBS	Suitable	Suitable	Allow	Need	Support	Support
PShED	Suitable	Unsuitable	Forbid	Need not	Support	Support
CBS	Limited	Suitable	Forbid	Need not	Don't support	Don't support
H-CBS	Limited	Suitable	Forbid	Need not	Don't support	Support

5 方法的共同特征分析及其有机集成

5.1 共同特征

在对不同方法的分析中,我们发现了几个重要的共同特征:

(1) 基于服务器方式,服务器模拟一定速度的处理机或其他全局共享计算资源.

(2) 以 EDF 算法作为系统调度器,负责服务器集合的调度,而每个服务器都用其截止期参数参与系统的调度.

(3) 具有处理机带宽隔离能力,如果与一个服务器相关联的就绪作业在执行中发生异常(如超出预计的执行时间),只会影响与该服务器相关联的其他作业的执行,而不会导致其他服务器关联的任务受干扰.

(4) 以固定的服务器速率执行与其相关联的就绪作业,并且每个服务器在它所声明的处理机带宽内执行,而不会在未被许可的情况下抢占其他服务器的带宽.

(5) 当系统的总处理机利用率≤1 时,各个服务器均可满足其时间约束要求.

5.2 有机集成的调度机制

通过考察以上共同特征,我们发现了在一个系统的调度机制中有机地集成各个方法的可行性.

根据共同特征(1),构造一个基于服务器方式的调度框架是可行的,所需做的是进一步融合不同类型的服务器.

根据共同特征(2)和(5),使我们可以在系统层使用一个EDF系统调度器,对由不同类型的服务器组成的服务器集合进行统一调度,并在总的系统处理机利用率≤1 的情况下,满足所有的服务器的时间约束.

根据共同特征(3)和(4),利用不同服务器的处理机带宽隔离能力,可为系统中同时存在的每个服务器(不论其类型)分配其所需处理机带宽,与各个服务器相关联的实时应用都被确保以一定的处理机带宽运行.另外,归属于不同服务器的任务之间的无关性,使系统级的异常隔离功能有了根本保证,即不同类型的各个服务器所关联的实时应用只需关心自身的异常处理,而不必担心其他实时应用的干扰.这也是开放式系统的主要需求之一.这样,不同类型的服务器上的实时应用不必作全局的实时性分析就可以满足其实时调度需求,符合开放式系统关于在同一系统中可独立开发和验证不同的实时应用的要求.

5.3 相对于现有方法的优越性

首先,本文前面提出的层次性调度对象模型覆盖了开放式实时系统涉及的全部调度对象,可用于集成的调度机制.与现有的基于单一任务模型的研究相比,可以方便地对开放式实时系统中不同层次的调度对象设计相应的调度方法.

其次,有机集成的调度机制对不同类型服务器的融合,使各类型服务器可以根据其自身的功能特征和适用范围,为相应类型的调度对象提供处理机执行时间,从而达到扬长避短的目的,以更好地支持多种类的实时应用的调度.

第 3,作为一个集成调度方法,它可以利用现有调度研究的成果,并允许研究者以它为框架增添新的服务器类型.当然,在可行性研究之后还需要进行进一步的方案设计.

6 结束语

开放式实时系统的相关研究是近年来的研究热点之一,其中的调度理论与方法是需要解决的关键问题.本文针对调度需求提出的层次性调度对象模型,相对于现有模型,更符合开放式实时环境的客观情况,覆盖了现有的所有实际调度对象.为研究更适合开放式实时系统的调度机制,我们对世界上现有的主要相关方法进行了深入的分析和比较,取得了有价值的成果:首先指出了各种现有方法的功能特征和适用范围,其次找出了它们的若干共同之处,而这些共同点可以启发我们有机地集成现有方法于系统的统一调度框架之中,并使它们各展所长.分析表明,有机集成的调度方法相对于现有方法具有多方面的优越性.对于具体的设计方案还有待于进一步的研究,这也是我们下一步的研究计划.

References:

[1] Stankovic JA., Spuri M, Ramamritham K, Buttazzo GC. Deadline Scheduling for Real-Time Systems-EDF and Related Algorithms. Boston: Kluwer Academic Publishers, 1998. 1~8.

[2] Liu JWS. Real-Time Systems. Upper Saddle River: Prentice Hall, 2000.

[3] Zou Y, Wang Q, Li, MS. The research and implementing of real-time support of Linux kernel. Journal of Computer Research and Development, 2002,39(4):466~472 (in Chinese with English abstract).

[4] Deng Z, Liu JWS. Scheduling real-time applications in open environment. In: Proceedings of the 18th IEEE Real-Time Systems Symposium. Los Alamitos, CA: IEEE Computer Society Press, 1997. 308~319.

[5] Parekh AK. A generalized processor sharing approach to flow control in integrated services networks [Ph.D. Thesis]. Massachusetts Institute of Technology, 1992.

[6] Kuo TW, Yang WR, Lin KJ. EGPS: a class of real-time scheduling algorithms based on processor sharing. In: Kelly K, ed. Proceedings of the 10th Euromicro Workshop on Real Time Systems. Los Alamitos, CA: IEEE Computer Society Press, 1998. 27~34.

[7] Deng Z, Liu JWS, Sun J. A scheme for scheduling hard-real-time applications in open environment. In: Proceedings of the 9th Euromicro Workshop on Real-Time Systems. Los Alamitos, CA: IEEE Computer Society Press, 1997. 191~199.

[8] Lipari G, Carpenter J, Baruah S. A framework for achieving inter-application isolation in multiprogrammed, hard real-time environments. In: Jacobs A, ed. Proceedings of the 21st IEEE Real-Time Systems Symposium. Los Alamitos, CA: IEEE Computer Society Press, 2000. 217~226.

[9] Abeni L, Buttazzo G. Integrating multimedia applications in hard real-time systems. In: Proceedings of the 19th IEEE Real-Time Systems Symposium. Los Alamitos, CA: IEEE Computer Society Press, 1998. 4~13.

[10] Lipari G, Baruah S. A hierarchical extension to the constant bandwidth server framework. In: Williams DA, ed. Proceedings of the 7th IEEE Real Time Technology and Applications Symposium. Los Alamitos, CA: IEEE Computer Society Press, 2001. 26~35.

附中文参考文献:

[3] 邹勇,王青,李明树.Linux 内核的实时支持的研究与实现.计算机研究与发展,2002,39(4):466~472.