支持多媒体应用的CPU 调度模型*

张拥军 陈福接

(国防科技大学计算机系 长沙 410073)

摘 要 多媒体应用的出现对操作系统调度模型提出了新的要求,许多多媒体系统中既有硬、软实时应用程序,又有传统的分时应用程序,希望在同一操作系统框架内得到支持。本文提出了支持上述功能的CPU调度模型,并设计了两种算法来实现该模型:基于QoS的CPU 带宽划分算法和双优先级调度算法。

关键词 多媒体应用,实时任务,CPU 调度模型

分类号 TP39

A CPU Scheduling Model Supporting Multimedia Application

Zhang Yongjun Chen Fujie

(Department of Computer, NUDT, Changsha, 410073)

Abstract The advent of multimedia application calls for new scheduling paradigms in operating system. The new paradigm should be able to handle the combination of hard real-time, soft real-time and conventional application present in many multimedia systems. This paper proposes a CPU scheduling model supporting the functions described above. The paper proposes two algorithms to implement this model: QoS-based CPU bandwidth partitioning algorithm and two-level priority algorithm.

Key words multimedia application, real-time task, CPU scheduling model

近年来,网络、数据压缩技术得到较大的发展,使得可视会议、VOD、虚拟现实等多媒体应用成为可能。在这些应用中,为了保证数字声、视频流表现的及时性和连续性,都对应用所基于的存储子系统、传输子系统和处理子系统提出了实时要求,希望系统为多媒体应用提供QoS(Quality of Service)保证,主要表现于系统的吞吐率、系统中的各种响应时间、系统的容错处理等方面,因此需要合适的操作系统的支持。

传统的分时计算系统和实时系统的实现采用了两种截然不同的方式,主要体现在两者所基于的操作系统。分时系统中操作系统使得多个应用共享系统资源,但不能确保各个应用执行的性能;实时系统通常需要专门的软硬件的支持,其实时操作系统能对有严格时间约束的任务调度执行。而目前随着多媒体技术在各个领域的广泛应用,如多媒体的监视控制系统、多媒体的信息查询系统等等,使得在多媒体应用系统中既有实时性要求的任务,又有普通任务(如分时执行的任务)。因此要在操作系统内核同时支持实时任务和分时任务的调度执行,也就是说要以合适的方式在各种多媒体应用任务之间分配CPU、内存,I/O 总线、网络带宽等。目前现有的商用操作系统都不能完全满足上述条件。

操作系统的任务调度往往涉及CPU、内存,还有网络、I/O 总线、磁盘、数据结构等资源,本文为了简化调度模型,只考虑关键元素CPU 的调度,提出了一个支持多媒体应用的CPU 调度模型,能在同一操作系统内集成若干种CPU 调度策略,同时支持实时任务和普通的分时任务的调度。针对应用类的不同,分别采用不同的调度策略,例如,用分时调度策略来调度分时任务的执行。本文也提出了实现该CPU 调度模型的两种算法:双优先级调度算法和基于QoS 的CPU 带宽划分法。该模型具有较好的扩

* 1997年9月11日收稿
第一作者:张拥军,男,1972年生,博士生

展性, 能根据应用类型的不同而加入相应的 CPU 调度策略。

目前国际上有许多研究机构也正在致力于同时支持实时应用与普通应用的操作系统的研究。如 Mach/RT Project[4]。另外, 人们也正在积极地把研究成果运用于商业操作系统, 如SunSoft 的Solaris2.1, IBM 的 OS/2等都具有了部分实时功能。

1 系统模型

根据多媒体应用系统中各种任务的要求来确定合适的 CPU 调度策略。针对实时任务、分时任务等, 通常有如下调度策略:

1.1 调度策略

硬实时任务

那些需要操作系统来确保各种响应时间的任务, 任务属性中与时间相关的参数, 如: 任务启动时间延迟、执行结束时间等都要由操作系统来预测保证。目前 EDF(Earliest Deadline First)[1] 和 RMA(Rate Monotoic Algorithms)[2] 是两种基本的实时任务调度算法, 都可用于硬实时任务的调度。

EDF: 优先调度具有最早执行结束时间约束的任务。该算法可用于调度动态实时系统: 系统中的任务可动态加入和撤离。

RMA: 是一种静态调度策略, 能较好地调度周期性实时任务的执行。

软实时任务

要求操作系统能统计地保证应用中与任务相关的 QoS 参数, 如最大延迟、错误率、吞吐率等。EDF、RMA 算法尽管可用于硬实时任务调度, 但算法要求事先知道各个任务的开始运行时间、运行周期、计算时间等参数。而在一些多媒体应用中, 如 VBR 视频, 通常对其计算的时间难以预计, 这些算法不适合软实时多媒体任务。文献[3] 中提出了 MSSA(Multimedia Server Scheduling Algorithms) 调度算法, 可用于软实时任务的调度。

普通任务

许多传统应用对时间没有严格的要求, 并不需要性能的保证, 这类应用中包括: 分时作业、后台作业等, 对这类应用, 操作系统通常采用分时调度算法或优先级调度算法。

优先级调度算法中每个任务都赋予严格的优先级, 按优先级次序从高到低执行, 分时调度算法是操作系统采用时间片流转的方法来执行系统中的各个任务。

从上可知, 针对不同性质的任务集, 往往采用不同的调度策略。为了支持多媒体应用中不同性质的任务集, 操作系统应能在同一调度框架集成不同的调度策略。

1.2 CPU 调度框架

CPU 调度模型如图1所示。

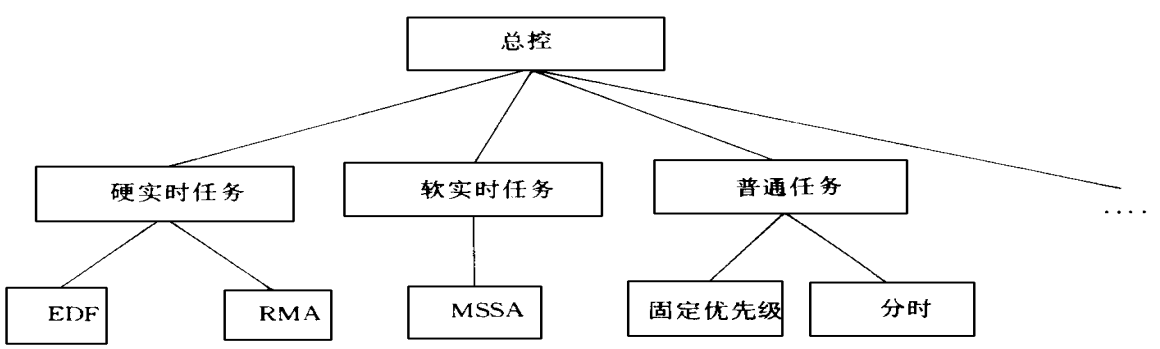

图1 CPU 调度模型

多媒体应用系统中的每个任务都属于树状模型(图1) 中的某个叶节点, 由叶节点代表的调度器来调

度执行。每个调度器分别采用相应的调度算法对其任务集(或任务队列)中的任务进行调度。如EDF调度器就对其任务集中的任务采用EDF算法进行调度。若某个调度器的任务集为空,则该调度器暂时不起作用。

图1中的每个中间节点对应一个应用类。图中列出软、硬实时,普通任务等三个应用类。每个应用类中的任务可根据任务的属性,如是否为周期性任务、是否为动态产生的任务等等,来细分。对细分后得到的任务集采用相应的调度算法。

图中的根节点是调度模型的总控模块,主要完成如下功能:

(1) 当系统中增加或删除应用类或调度器时,在总控模块登记,对调度器进行管理。

(2) 当系统中有新任务加入时,将任务根据其属性送往某个调度器,由其调度执行。

2 实现算法

针对设计的操作系统CPU调度模型,提出两种实现算法。

2.1 双优先级调度算法——一种改进的优先级调度方法

实时应用的基本要求就是要调度器能在满足时间约束的条件下调度其执行。而当系统中有多种类型的应用(如硬、软实时应用,普通应用)时,由于它们对时间的敏感程度不一样,因此导致在它们之间进行的CPU时间分配也会出现不公平现象。在优先级调度算法中,高优先级的任务推迟低优先级任务的执行,这是可接受的。因此当系统中出现若干类应用时,可赋实时任务高优先级,保证其能被抢先调度;而常规任务可赋较低的优先级。然而在同一个系统中,将不同类的应用任务在单一的一个优先级队列中分配优先级,不利于系统的扩展,如增加一个新的应用类,希望新类中的任务能先于一般普通应用任务执行,而又不能与实时任务抢占CPU,这时在一个优先级已排好的系统中,很难为新应用类中的任务赋优先级。因此,提出一种改进方案:采用双优先级调度算法。将过去传统的单一的优先级表示转化为双优先级表示:

(CPU调度策略,优先级)

系统先对已知的应用进行划分,为每个应用类选取相应的CPU调度策略。然后根据应用类中的任务对时间的敏感程度来对CPU调度策略排序。因此,系统中的每个任务先按它们基于的调度策略排序,通常实时任务的调度策略的优先级高于其它类的任务。在每个与调度策略相对应的任务集中,再给任务赋予一定的优先级,如实时任务中,有两种调度策略EDF和RMA,由EDF算法调度的任务按预定的执行结束时间来排序,赋予调度优先级,由RMA算法调度的任务可按任务周期的长短来定优先级。因此这种方法中,调度策略优先级高的任务优先被调度执行。任务的双优先级表示法有利于系统应用的扩充。增加一个新应用类,只要相应增加调度策略,并将其安排在合适的优先级位置就可以,而不用调整其它应用类中任务的优先级值。

运行模型:

根据双优先级调度算法的思想,操作系统为每种调度策略组织一运行队列。当操作系统要作CPU调度时(如系统中增加新任务,任务主动放弃CPU),操作系统按调度策略的次序,启动相应的调度算法来调度相应的运行队列中的任务。如图2所示。

2.2 基于QoS的CPU带宽划分法

在图1基础上,增加一个QoS管理器,如图3所示。由QoS管理器在各类应用任务间划分CPU资源。

(1) 系统中各个任务向QoS管理器说明各自QoS的要求,由QoS管理器根据任务所属的应用类来计算各个任务需要的资源。从而从下至上统计出各类应用所需的CPU资源。

(2) 在过程1中,如果应用为硬(软)实时任务,那么QoS管理器用确定性(统计性)允许控制算法[1, 2, 5]来判断应用请求的CPU资源能否被满足。若不能满足,则该任务请求被拒绝。

(3) 在过程1中,如果应用为普通任务,如分时任务等,那么任务无条件接收。

(4) QoS管理器根据在过程(1)中计算出来的结果,在各类应用间划分CPU带宽。

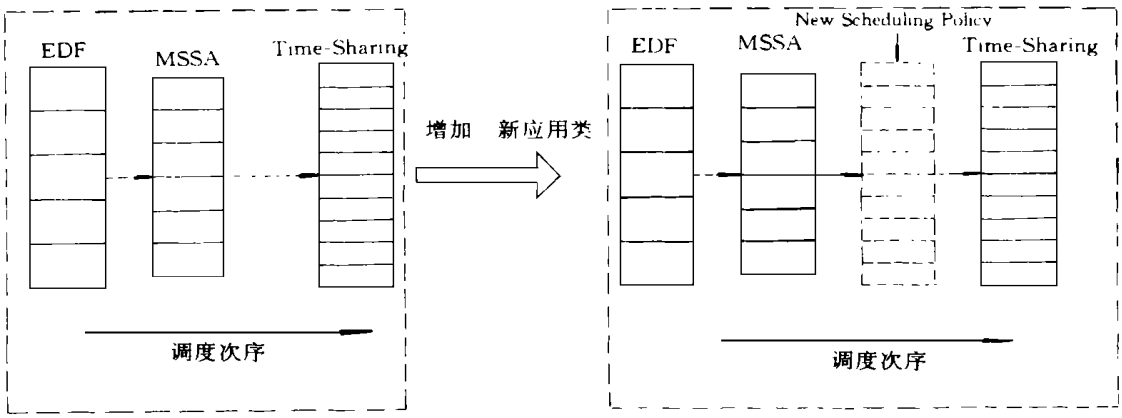

图2

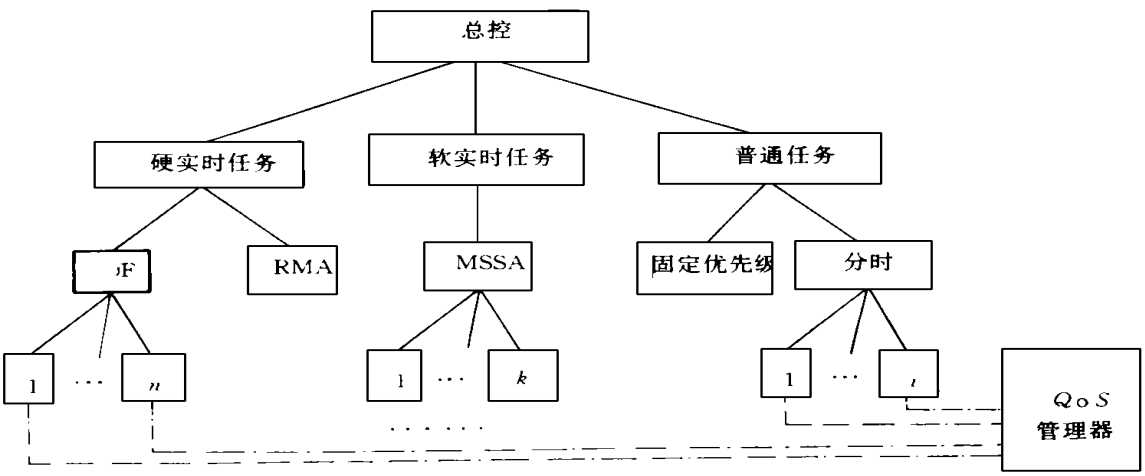

图3

为树(图3)的非叶节点计算权重。先对根的各个子节点分配权重,这一级各节点之间权重之比,大致体现各类应用占用 CPU 带宽的情况。由 QoS 管理器计算得到的各类应用需要的资源情况,来分配权重。也就是在几个大的应用类之间分配 CPU 带宽,在图3中,若在硬实时、软实时和普通任务之间由 QoS 管理器计算得到的权重之比为2XX5,那么相应地,它们占用的 CPU 带宽之比也为2XX5. 然后往下,在各类应用的子类中继续划分 CPU 带宽。硬实时应用划分为两类: 分别由 EDF 和 RMA 算法调度的任务。相应地,整个硬实时应用分到的 CPU 带宽继续在这两个子类中划分。

应用之间 CPU 带宽划分是可动态改变的。若系统中无硬实时任务,那么软实时与普通应用之间的 CPU 带宽划分就为3X5. 若系统中软实时任务增多,由 QoS 管理器计算,相应调整各类应用之间的权重之比,使软实时任务占用的 CPU 带宽适当提高。

根据节点的权重,节点(非叶节点)占用 CPU 带宽的计算方法如下:

假设第 i 级 $n_1, n_2 \cdots n_r$ 是某个 $i-1$ 级节点 N 的 r 个子节点。$w_1, w_2, \cdots, w_r$ 是 r 个节点所对应的权重,那么,若节点 N 分配的 CPU 带宽为 B,则其 r 个子节点分配带宽分别为:

$$B_i = \frac{w_i}{\sum_{j=1}^{r} w_j} B, \quad i = 1, 2, \cdots, r.$$

假设根节点 CPU 带宽已知,为 B_{root}.

任务的调度。CPU 带宽的划分只到调度器一级(图3中标有 EDF、RMA、MSSA 等调度策略的节

点)。每个调度器在其划分到的带宽内对其任务集中的任务进行调度运行。为了不出现混乱情况, QoS 管理器还要对 CPU 带宽的使用进行管理, 安排各调度器的运行。

系统的扩展。当系统中增加新的应用时, 只要 QoS 管理器根据新应用中任务 QoS 要求, 计算各任务所需资源, 统计出新应用需要的 CPU 资源, 再根据整个系统 CPU 使用情况, 赋予适当的权重, 来划分 CPU 带宽。

上面阐述的双优先级方法和基于 QoS 的 CPU 带宽划分法, 是在第2节中提出的支持多媒体应用的 CPU 调度模型的具体实现, 都具有较好的扩展性。双优先级法简单, 易于实现, 但相对来说各类应用之间 CPU 时间分配不太公平, 可能出现普通任务被饿死的情况。基于 QoS 的 CPU 带宽划分法相对复杂些, 但 CPU 资源的分配相对公平。

3 结束语

本文提出了一个同时支持硬、软实时, 分时等应用的 CPU 调度模型, 以满足多媒体应用的需要。在该模型中可集成若干种不同的调度策略, 可以支持多种应用。本文也提出了两种实现该模型的方法: 双优先级算法和基于 QoS 的 CPU 带宽划分法。

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, 20(1)

2 Lehoczhy J, Sha L, Ding Y. The Rate Monontonic Scheduling Algorithm-Exact Characterization and Average Case Behavior In Proc. of the Real-time System Symposium, 1989: 166～171

3 Kaneko H, Stankovic J A. A Multimedia Server on the Spring Real-time Systems Umass Computer Science Technicial Report 96- 11, January 1996

4 Shipman, Samuel, Teller. Mach/RT kernal Interfaces Technical Report TR92- 11, Center for High performance computing, Worcester Polytechic Institute. 1992

5 Stankovic J A, Spuri M, Natale M D. Implications of Classical Scheduling Results fot Real-Time Systems Umass Computer Science Technicial Report 94- 89, 1994