基于嵌入式RTOS的闭环反馈调度算法的研究

王 铮,孙 萍
(重庆大学 计算机学院,重庆 400044)

摘 要:分析了常见调度算法的特点,提出一种基于嵌入式实时操作系统的闭环反馈控制调度策略。针对任务的特点,从任务的周期与非周期性、I/O消耗和处理器消耗型两个方面对任务进行相应的反馈控制,使调度器具有一定程度上的自适应功能。并对实时操作系统μC/OS-II的内核调度算法进行改进,同时与EDF算法进行对比测试,可以看出改进后的调度算法在系统负载较重或系统过载的情况下表现良好。

关键词:嵌入式实时系统;调度算法;反馈控制;自适应调度
中图分类号:TP301.6 文献标识码:A 文章编号:1673-629X(2007)11-0026-04

A Closed-Loop Feedback Scheduling Algorithm Based on Embedded Real-Time Operating System

WANG Zheng, SUN Ping
(College of Computer Science and Technology, Chongqing University, Chongqing 400044, China)

Abstract: Analyses common characteristics of scheduling algorithms, and presents an embedded real-time operating system which is based on a closed-loop feedback control scheduling strategy. In view of the characteristics of task, periodic and aperiodic, I/O-bound and processor-bound tasks, the two aspects of tasks carries on corresponding feedback control, which makes the process scheduler has certain procedures for scheduling of adaptive function. And compared with EDF, the modified scheduling algorithm of μC/OS-II performs good under the situations when the system is in heavy load or overload.

Key words: real-time operating system; scheduling algorithm; feedback control; adaptive scheduling

0 引 言

嵌入式系统是针对完成特定功能而设计的软硬件结合并可裁剪的专用计算机系统。实时操作系统是指能够进行实时处理的一类操作系统。实时性是嵌入式系统性能的重要指标之一。

自1981年世界上第一个商业嵌入式实时内核(VRTX32)问世到今天已经有20多年,嵌入式实时操作系统RTOS(Real Time Operating System)在嵌入式系统中的主导地位已经确定。目前代表性产品有VxWorks, QNX, Lynx和WinCE等,由于嵌入式应用本身具有的多样性,其对应的操作系统很难垄断,只要在某个应用领域成功就算成功,因此不会出现一个标准的RTOS。在操作系统中,任务调度策略是内核设计的关键部分,如何采用适当的算法来保证各个任务都能在其期限之内完成是实时操作系统研究的一个重要领域。

经典调度算法在系统配置的约束下存在一些实际的问题,如何在处理能力有限的条件下,满足系统实时性是一个迫切需要解决的问题。

1 实时调度

实时任务与一般任务之间的最大不同就在于要满足处理与时间的关系。由于实时系统的侧重点不同,实时调度也有多种分类方式。

1.1 实时调度的分类

根据对实时性能要求的程度,实时任务可分为硬实时和软实时两类;根据调度顺序产生的时机和方式可分为静态调度和动态调度;根据任务到达时刻规律的不同,可分为周期任务调度和非周期任务调度;根据调度方法是否具有自适应功能,可分为自适应调度和非自适应调度。

1.2 常见的实时调度算法

实时调度模型随着对实时系统认识的加深而不断变化。硬实时周期任务的调度模型由C. L. Liu和J.

收稿日期:2007-02-02
作者简介:王 铮(1953-),男,重庆人,副教授,研究方向为软件自动化、嵌入式系统。

Layland 在 1973 年提出[1]。而基于该硬实时周期任务的调度算法 RM (Rate Monontic), DM (Deadline Monontic)则被广泛应用于工业控制领域。由于硬实时周期任务的特征明显, 相应的研究已进行得比较透彻。相比之下, 非周期软实时任务由于种类繁多, 不易分类, 相应的调度模型仍然是实时调度研究中的重点。

常见的实时调度算法有速率单调算法(Rate Monotonic, RM)、时限单调算法(Deadline Monotonic, DM)、最早时限优先调度算法(Earliest Deadline First, EDF)、最小空闲时间优先调度算法(Least Laxity First, LLF)。也有其它的一些在这些算法上优化或者修改而成的算法, 如改进的最小空闲时间算法等。

不同的算法有不同的适应条件和相应的局限性。表 1 列出了其具体的优缺点。

表 1　常见的实时调度算法特点对比

RM	优点	系统开销小, 是最优静态优先级调度算法, 适用于高安全性周期性任务
	缺点	CPU 利用率低, 不灵活, 缺少对突发事件实时处理能力
EDF	优点	动态优先级策略中的最优算法, 实现很高的 CPU 利用率, 效率高, 容易计算和推断
	缺点	系统开销大, 很难诊断出即将过载的可能性, 比较适用于软实时系统
LLF	优点	在系统重负载的情况下工作较好
	缺点	颠簸现象较为严重, 增大了系统开销

2　闭环反馈控制系统

近几年, 作为在不可预测环境下获得控制性能保证的经济有效方法, 闭环反馈控制系统[2]在调度算法中得到了发展与应用。

闭环反馈控制系统是指根据输入与输出的实际情况来决定控制策略, 以便达到预定的功能。通常操作系统中使用的闭环进程调度为负反馈控制, 即根据所希望的系统性能与实际输出的性能指标之间的差值作用于系统调度器, 进行系统的动态调整。闭环反馈控制系统框图如图 1 所示。

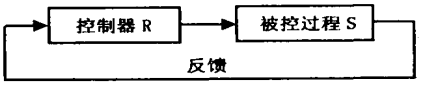

图 1　反馈控制系统框图

3　闭环反馈控制调度算法的设计与实现

3.1　闭环反馈控制设计思想的两个方面

文中所研究的闭环反馈控制调度策略主要从以下两个方面来体现。

3.1.1　周期与非周期任务的处理

在实时系统中, 任务的启动与外部环境密切相关。根据外部驱动源不同, 实时任务可分为两类: 周期任务和非周期任务。周期任务的发生一般有固定的周期; 非周期任务大多由外部事件驱动。由于事件的发生是随机的, 这类任务的到达时间也是随机的。

在反馈控制策略中, 对周期任务采用 RM 调度算法; 而对于非周期任务, 则采用 FIFO 调度算法。

3.1.2　I/O 消耗和处理器消耗型任务的处理

任务可以被分为 I/O 消耗型和处理器消耗型两类。前者指任务的大部分时间用来提交 I/O 请求或是等待 I/O 请求。因此, 这样的任务经常处于可运行状态, 但一般都只运行短短的一段时间。对这类任务来说, 调度策略希望适当提高它们的优先级, 减少其运行时间。相反, 处理器消耗型任务将时间大多用在执行代码上。除非被抢占, 否则它们通常都一直不停地运行。对于这类任务, 调度策略总是希望降低它们的运行优先级, 延长其运行时间。

3.2　闭环反馈控制调度原理

文中提出一种基于动态优先级的闭环反馈调度策略, 其系统框架图如图 2 所示。其中, A_i 代表优先级为 i 的周期性任务列表, S_i 代表优先级为 i 的非周期性任务列表, T_k 代表优先级为 k 的任务列表, $T_{(n-2)p}$ 代表优先级为 $(n-2)$ 的任务列表中的第 p 个任务。

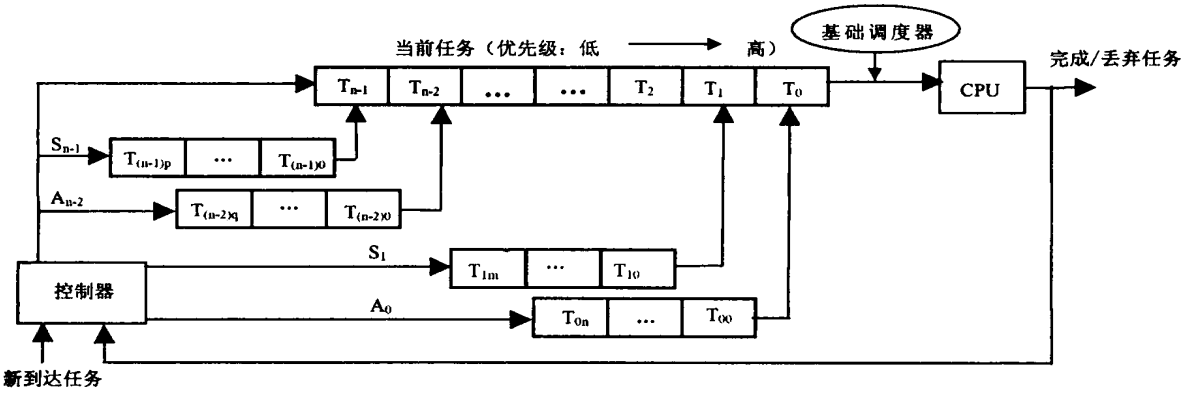

图 2　闭环反馈控制调度算法系统框架图

当前任务队列中的 n 个优先级的每位优先级对应一个任务列表(如: T_i 的任务列表为: T_{i0} 到 T_{in})。在这个列表中,每个任务都用一个字节 pribyte 来代表其二次优先级, T_{in} 中 n 值越小,其二次优先级越高,当该优先级运行时,在其对应队列中选择 n 值最小的任务运行即可。

将连续的两个优先级划为一级, n 个任务可分成 $n/2$ 个等级(n 为偶数),每个等级中偶数位代表周期性任务列表,对其中的任务采用 RM 调度算法;奇数位则代表非周期性任务列表,对其中的任务采用 FIFO 调度算法。以此来实现对周期与非周期任务的不同处理。如果是新任务到达,默认是非周期性任务。

新到达任务有一个基本的优先级,本调度程序允许控制器根据任务类型(I/O 消耗型和处理器消耗型)的需要来适当地加、减优先级。同时对应于当前任务中的 n 个优先级,每个优先级任务列表的时间片均不相等。高优先级设计时间片短,为 5 毫秒;低优先级设计时间片长,为 200 毫秒,其它优先级的时间片在此间成等差数列。每个任务都记录自身用于休眠和用于执行的时间。用于执行的时间存放于任务结构体的 runtime 域中。它的范畴从 prio_ runslice(优先级为 prio 的时间片长度)到 0,当任务从休眠恢复到执行状态时,runtime 会根据运行的时间长短减少,直到小于 0。同时,任务每运行一个时钟节拍,如果上次 runtime 小于 0,则 task_ runprio_ number 就作相应的增加,当该值大于 3 就表示这个任务已经连续三次运行时均用完了时间片,刚将其优先级加 2;反之,则优先级减 2。当一个任务优先级的改变量已超过正负 10 后,则优先级不再改变。

这种推断机制不仅会奖励交互性强的任务,它还会处罚处理器耗费量大的任务,并且反应迅速。

3.3 调度策略在 μC/OS—II 内核中的实现

μC/OS—II 系统采用 64 位静态优先级分配策略,在程序运行前由用户为每个任务指定优先级[3]。

3.3.1 控制器

在这里用尽可能简便的方法判断出该任务是否是周期任务,为了支持这种判断机制,每个任务都记录了它的到达时间,存放于任务结构体的 time 域中。当任务再一次就绪时,利用此时的时间就可算出两次就绪时间的间隔 space,利用连续几次的间隔时间来判断是否是周期任务。

对于任务 T_x(x 为任务的优先级),控制器对其周期性处理的算法流程图如图 3 所示。任务依据 RM 算法归入 A_i 队列的算法流程图,如图 4 所示。归入 S_i 队列中,用 FIFO 算法插入。

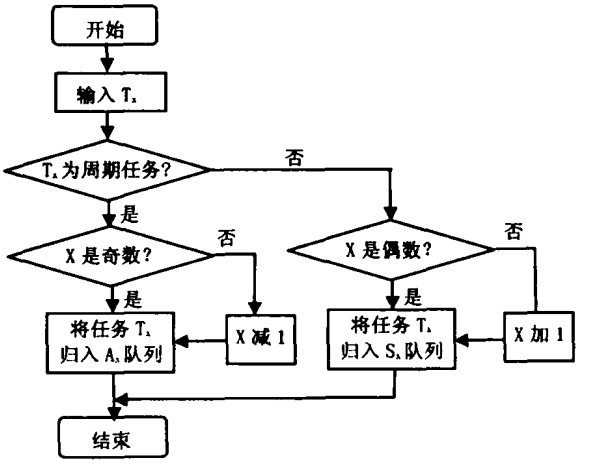

图 3 控制器对周期与非周期任务处理算法流程图

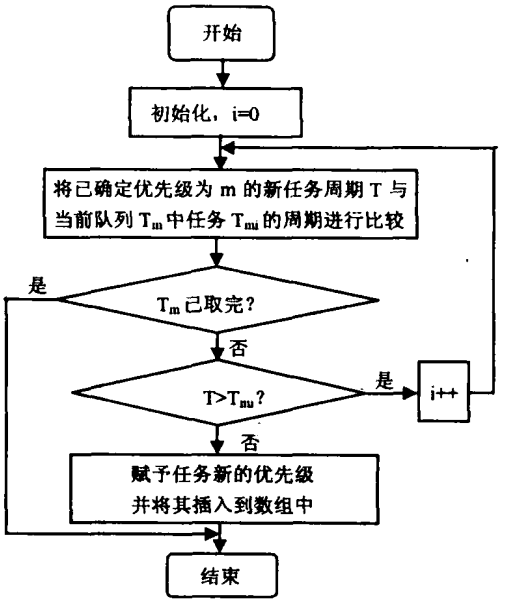

图 4 RM 算法流程图

3.3.2 基础调度器

定义调度管理结构体如下:

```
struct ScheduleStr
{
int tasknum;
unsigned byte bitmap[4];
struct list_head queue[64];
}
```

在调度管理结构体 ScheduleStr 中选定下一个优先级最高的进程,只需要判定 bitmap 中优先级最高的位 i,并选择运行 T_{i0} 即可[4]。具体执行过程可调用 μC/OS—II 自身的 schedule()函数实现。

4 新调度器仿真与测试

在 μC/OS—II 上改进调度器的调度算法,首先要搭建一个基于 X86 微处理器的 PC 机平台。在这里采

用 Borland C/C++ V4.5 的C 编译器和 Borland Turbo Assembler 汇编器完成程序的移值和测试[5]。

4.1 仿真测试方法

每个任务的最坏执行时间在 200ms 内均匀随机地选择。任务的周期 T_i 按泊松分布:

$$T_i = \frac{N * C_i}{p}$$

其中 T_i 表示任务 i 的周期, N 表示总任务数, C_i 表示任务 i 的最坏执行时间, p 表示期望产生的工作负载。

4.2 仿真结果

为了测试改进后的闭环反馈控制调度器的性能, 将 EDF 调度算法和文中的反馈控制调度算法仿真测试结果进行对比, 主要比较两种算法的上下文切换次数和截止期错失率。

①上下文切换次数的比较。

取 $N = 20$, p 从0.7 到 1.3, 分别对两种算法进行调度仿真, 比较在调度仿真器发生 1000 次中断时, 两种调度算法的上下文切换次数。如图 5 所示。

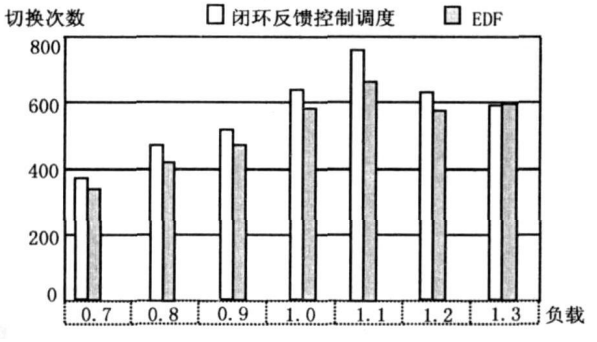

图 5　闭环反馈调度与 EDF 上下文切换次数

②截止期错失率的比较。

取 $N = 20$, p 从 0.7 到 1.3, 分别对两种算法进行调度仿真, 比较在调度仿真器发生 1000 次中断时, 两种调度算法的截止期错失次数。如图 6 所示。

由图 5 和图 6 可知, 在任务切换方面, 反馈控制调度算法不如 EDF 算法, 但在系统负载较重时, 反馈控制调度算法可以从一定程度上调节任务优先级, 使上下文切换次数反而小于 EDF 算法。在任务错失率方面, 当系统负载小于1时, EDF算法截止期错失次数为 0; 当系统负载大于 1 时, 反馈控制算法错失率小于 EDF 算法, 并且当系统过载越大时, 这种差距越明显。

由此可知, 本反馈控制调度算法适用于系统负载较重或系统过载的情况。

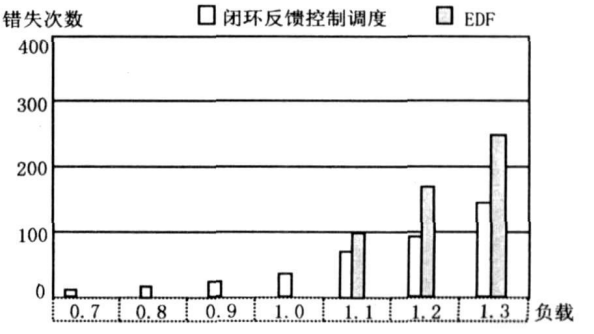

图 6　闭环反馈调度与 EDF 截止期错失次数

5 结束语

调度算法是嵌入式实时系统内核的关键组成部分。文中的反馈控制调度算法将自动控制理论和调度算法结合在一起, 建立了实时系统的调度模型, 并对改进后的调度算法进行了测试。测试表明该调度器在系统负载较重或系统过载时具有良好的调度性能, 可以满足大部分工业控制系统实时要求。能够在更广的范围内使任务得到及时的调度, 充分发挥嵌入式系统的性能。

参考文献:

[1] Liu C L, Layland J. Scheduling Algorithms for Multiprogramming in a Hard—Real—Time Environment[J]. Journal of the Association for Computing Machinery, 1973, 20(1): 46—61.

[2] 童立靖. 实时系统的自适应进程调度方法研究[D]. 北京: 中国科学院研究生院, 2004: 7—25.

[3] 邵贝贝. 嵌入式实时操作系统μC/OS—II[M]. 第2版. 北京: 北京航空航天大学出版社, 2003: 88—103.

[4] Love R. Linux 内核设计与实现[M]. 第2版. 陈莉君, 康华, 张　波, 译. 北京: 机械工业出版社, 2006: 29—49.

[5] 魏立峰, 于海斌. 一种基于自适应控制的软实时调度算法研究[J]. 系统仿真学报, 2004(4): 760—771.

(上接第 25 页)

参考文献:

[1] 李笑歌, 杨　扬, 景云华. 基于视频图像的车辆自动分类系统[J]. 微计算机信息, 2003, 19(8): 38—39.

[2] 蔡晋辉, 周泽魁. 机器视觉系统在桔瓣质量检测中的应用[J]. 农业工程学报, 2004, 20(6): 129—132.

[3] 应义斌, 景寒松, 马俊福. 用计算机视觉进行黄花梨果梗识别的新方法[J]. 农业工程学报, 1998, 14(2): 221—225.

[4] 章旒晋. 图像处理和分析基础[M]. 北京: 高等教育出版社, 2002.

[5] 付忠良. 图像阈值选取方法的构造[J]. 中国图像图形学报, 2000, 5(6): 466—469.

[6] 孙即详. 图像分析[M]. 北京: 科学技术出版社, 2005.