

一种新的嵌入式 Linux 实时调度算法

霍 华, 崔欢欢

(河南科技大学 电子信息工程学院, 河南 洛阳 471003)

摘 要: 针对现有实时调度算法在实时性能上的不足, 提出了一种新的实时任务分类调度算法, 根据实时任务的到达情况进行分类, 划分为周期任务和非周期任务, 并对不同类型的实时任务使用不同的改进的实时调度算法. 通过实验, 证明此算法与现有实时调度算法相比, 综合实时性能有了很大程度的提高.

关键词: 嵌入式; 实时调度算法; 分类调度算法; 周期任务; 非周期任务

中图分类号: TP391

文献标识码: A

文章编号: 1000-7180(2011)06-0138-04

A New Real-time Scheduling Algorithm for Embedded Linux

HUO Hua, CUI Huan-huan

(Electronic Information Engineering College, Uni. of Sci. & Tech., Luoyang 471003, China)

Abstract: To solve the insufficiency of real-time performance in the present real-time scheduling algorithm, a new real-time task classification scheduling algorithm is proposed. According to the arrival of real-time tasks are classified, and divided into periodic tasks and non-periodic tasks, and use different improvement real-time scheduling algorithm to schedule different type of real-time task. Experiments show that classification scheduling algorithm is compared with the present real-time scheduling algorithm, has been largely improved for integrated real-time performance.

Key words: embedded; real-time scheduling algorithm; classification scheduling algorithm; periodic task; non-periodic tasks

1 引言

Linux 以源码开放、系统稳定等优势, 在嵌入式系统、实时控制等领域得到广泛应用. 但 Linux 属于通用的分时操作系统, 实时性能较差, 因此需要对其进行改进以应用实时环境^[1]. 实时调度算法是影响系统实时性能的重要因素之一^[2], 根据调度驱动的机制, 实时调度算法可分三种类别^[3]: 时间驱动的调度算法 (Time-Driven scheduling, TD)、优先级驱动的调度算法 (Priority-Driven scheduling, PD) 和共享驱动的调度算法 (Share-Driven scheduling, SD).

优先级驱动的调度算法是最流行的调度算法, 其原理是给每个任务分配一个优先级, 在每次任务调度时选择具有最高优先级的就绪任务来执行. 它有静态和动态优先级调度算法两种类型, 典型的代

表算法为 RM 和 EDF 调度算法^[4], 还有一些在此基础上改进的算法, 如 NSRL^[5]、LLF^[6]等.

目前采用在嵌入式 Linux 上增加实时调度算法的方式来提高嵌入式系统的实时性能, 但这种方式在调度混合实时任务时存在缺陷, 很多调度算法在调度包含周期任务和非周期任务的混合任务集时, 仅是简单地将非周期性任务看作是特殊的周期性任务, 使得实时性能大幅度下降. 实际上根据任务不同的性质, 实时任务可以分为不同的种类^[7], 例如: 根据任务的到达情况, 任务可分为周期任务和非周期任务, 周期任务是指按一定周期到达并请求运行的任务, 非周期任务是指随机到达系统的任务. 本文采用一种新的实时任务分类调度算法, 对周期任务和非周期任务分别使用静态和动态调度算法, 最大程度地提高系统实时性能.

收稿日期: 2010-09-20; **修回日期:** 2010-11-30

基金项目: 国家自然科学基金项目 (60743008); 河南省科技攻关计划项目 (072102210010)

2 分类调度算法

分类调度算法的总体架构如图1所示,使用动态调度算法来调度非周期性实时任务,静态调度算法来调度周期性实时任务。

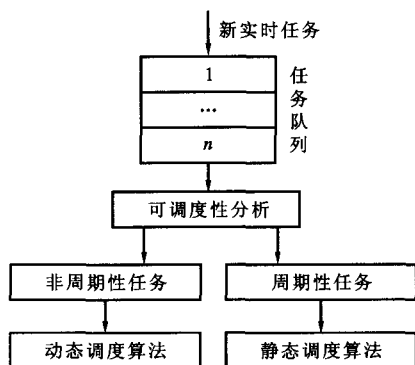


图1 分类调度算法思想

分类调度算法的分配步骤如下:

(1) 对新到达的实时任务进行可调度性分析,若是非周期性任务,则执行步骤(2),否则执行步骤(3)。

(2) 对新到达的非周期性实时任务,进行动态调度算法的可调度性分析,若符合可调度条件,则该实时任务是可调度的,否则不可调度。

(3) 对新到达的周期性实时任务,进行静态调度算法的可调度性分析,若符合可调度条件,则该实时任务是可调度的,否则不可调度。

2.1 任务模型

在单处理器环境下,定义一个包括 n 个实时任务的集合 $T = \{t_1, t_2, \dots, t_n \mid (1 \leq i \leq n)\}$,规定:

(1) 集合中非周期性实时任务的时间属性可以用一个三元组 $t_i = (A_i, C_i, D_i)$ 来表示。其中, A_i 表示任务的到达时间, C_i 表示任务的最坏执行时间, D_i 表示任务的相对截止期,规定任务必须在绝对截止期 $d_i = A_i + D_i$ 之前完成。

(2) 集合中周期性实时任务的时间属性可以用一个三元组 $t_i = (A_i, C_i, P_i)$ 来表示。其中, P_i 表示任务的周期。

(3) 各任务之间相互独立。

(4) 任务具有可抢占性,较高优先级的任务可以抢占较低优先级任务的 CPU 控制权。

2.2 相关定义

定义1 处理器利用率 (processor utilization): 实时任务在执行时花费处理器时间的比例,即最坏执行时间与活动时间之比。实时任务 t_i 的处理器利用率为 C_i/D_i (t_i 为非周期性任务) 或

C_i/P_i (t_i 为周期性任务)。

实时任务集的处理器利用率为实时任务集中所有实时任务的处理器利用率之和。执行 n 个实时任务的处理器利用率 U 如式(1)、(2)所示:

$$U = \sum_{i=1}^n \frac{C_i}{D_i} \quad (t_i \text{ 为非周期性任务}) \quad (1)$$

$$U = \sum_{i=1}^n \frac{C_i}{P_i} \quad (t_i \text{ 为周期性任务}) \quad (2)$$

定义2 空闲时间 (Laxity Time): 实时任务的空闲时间 L_i 为任务的绝对截止期减去任务的剩余执行时间 E_i 再减去当前时刻 t , 如式(3)所示:

$$L_i = d_i - E_i - t \quad (3)$$

定义3 最坏执行时间 (Worst Case Execution Time): 最坏执行时间 C_i 是指实时任务执行时所需时间的最大值,包括调度任务的多种时间开销。

定义4 重要程度 (Importance): 是一个正整数,说明此任务与其他任务相比的重要性。

3 常用调度算法分析

(1) 速率单调 (Rate-Monotonic, RM) 调度算法,是一种典型的静态优先级调度算法,根据任务周期的长短来决定调度优先级的高低,较小的任务周期对应较高的优先级。优点为:实现机制简单和调度开销较低。缺点为:仅以任务周期作为优先级的判定条件,容易忽视那些比较重要但周期较长的任务;CPU 利用率低;资源不够灵活,对突发事件实时处理的能力不强。

(2) 最早截止期优先 (Earliest Deadline First, EDF) 调度算法^[8],是一种典型的可抢占的动态优先级调度算法,根据各个任务的截止期限来分配优先级,具有最早截止期限的任务具有最高的优先级。优点为:资源分配灵活、保证截止期最早的任务的优先执行和 CPU 利用率高。缺点为:系统开销大;不能保证不可达截止期任务的执行;超载情况下会发生“多米诺”现象使实时性能急剧下降。

(3) 最少空闲时间优先 (Least Laxity First, LLF) 调度算法,也是一种常用的动态优先级调度算法,是对 EDF 算法的改进,根据任务空闲时间的非降顺序动态分配优先级,空闲时间较短的任务具有较高的优先级。规定正在运行的任务的空闲时间保持不变,而就绪队列上的任务的空闲时间随时间递减,优先级也随之动态的改变。优点为:节省 CPU 资源和保证紧急任务的优先执行。缺点为:等待任务的优先级可能会随空闲时间的减小而变高,抢占当

前正在运行的任务,造成任务之间的频繁切换(即颠簸)现象,使系统性能大幅度下降。

4 改进调度算法分析

4.1 动态调度算法

根据以上常用算法的分析,对非周期性实时任务,采用一种改进的 DAL (Deadline And Laxity First) 调度算法,它综合考虑了截止期限和空闲时间对优先级的影响,并增加抢占阈值的调度算法来减少颠簸。

4.1.1 DAL 调度算法分析

该算法结合了 EDF 算法和 LLF 算法各自的优点,优先级 P_i 的大小,根据式(4)而动态发生改变:

$$P_i = \alpha * d_i + (1 - \alpha) * L_i \quad (4)$$

式中, α 是实时任务的平衡因子,大小决定截止期限和空闲时间对优先级的影响程度. α 为 0 时倾向于 EDF 算法,为 1 时倾向于 LLF 算法。

抢占阈值的调度算法的思想是^[9]: 利用优先级对 (P_i, Pr_i) 来控制任务的执行,其中 P_i 为基本优先级, $Pr_i (P_i < Pr_i < n)$ 是任务的抢占阈值,规定允许抢占的任务的基本优先级同时满足,大于被抢占任务的基本优先级和抢占阈值. 吸收了抢占和非抢占式调度算法各自的优点,减少了任务间的抢占次数,降低了颠簸带来的系统开销. 描述任务调度的完整算法步骤如图 2 所示。

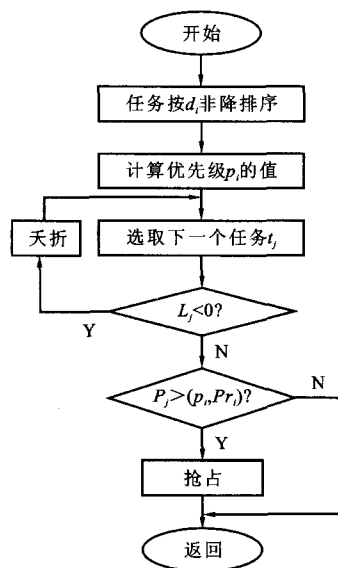


图2 算法的任务调度流程图

4.1.2 算法可调度性分析

对于 n 个相互独立的非周期性实时任务集,处理器利用率的最小上限是 1. 因此,只要实时任务集的处理器利用率 $U \leq 1$,则它们是可调度的。

当一个新的非周期性实时任务 $t_i(A_i, C_i, D_i)$ 到达时,如果满足式(5):

$$\frac{C_i}{D_i} + U_n \leq 1 \quad (5)$$

那么它是可调度的. 其中, U_n 为当前实时任务集的处理器利用率。

4.2 静态调度算法

根据以上常用算法的分析,对周期性实时任务,采用一种改进的 RAI (Rate And Importance) 调度算法,综合考虑了周期和重要程度两个因素在优先级中所占的比重,提高了重要任务的执行概率。

4.2.1 RAI 调度算法分析

在 RAI 算法中,任务的优先级在运行前指定,由周期和重要程度共同决定,其中重要程度由用户根据需求自己设定. 规定:周期短的任务,优先级高;重要程度高的任务,优先级高。

该算法可分为以下几种情况:

(1) 周期相同、重要程度不同的实时任务,按照重要程度由高到低的顺序执行。

(2) 重要程度相同、周期不同的实时任务,按照 RM 调度算法,周期较短的任务优先执行。

(3) 周期和重要程度完全相同的实时任务,按照 FIFO 的方式进行调度。

(4) 周期和重要程度都不同的实时任务,以重要程度的高低作为抢占的条件,保证重要任务的优先执行。

4.2.2 算法可调度性分析

对于 n 个相互独立的周期性实时任务集,处理器利用率的最小上限是 $n * (2^{1/n} - 1)$. 因此,只要实时任务集的处理器利用率 U 不大于最小上限,则它们是可调度的。

当一个新的周期性实时任务 $t_i(A_i, C_i, P_i)$ 到达时,如果满足公式(6):

$$\frac{C_i}{P_i} + U_n \leq n * (2^{1/n} - 1) \quad (6)$$

那么它是可调度的. 其中, U_n 为当前实时任务集的处理器利用率。

5 性能测试与分析

为了测试本文算法的实时性能,在 AMD Sempron 2.01 GHz 单处理器和 1.75 GB 内存的实验平台下,对包含 100 个任务的混合实时任务集,分别采用本文算法、EDF 调度算法、RM 调度算法,以调度延迟和截止期限失率为性能评价指标进行了测试。

测试中,任务的提交顺序一样,前后任务提交时间相差 1 s.

5.1 调度延迟

调度延迟 (Scheduling Latency, SL) 表示实时任务从调度机会发生到被调度所花费的时间,是衡量实时性能的一个重要指标. 调度延迟越低,实时性能越好.

从表 1 中可以看出,本文算法无论最小、最大、平均调度延迟都小于 EDF 调度算法,但略高于 RM 调度算法. 说明本文算法的调度延迟性能介于 EDF 调度算法与 RM 调度算法之间,调度开销比 RM 调度算法稍高,而远低于 EDF 调度算法.

表 1 调度延迟测试结果对比

方法	调度延迟/ μs		
	最小	最大	平均
EDF 调度算法	6	19	13.92
分类调度算法	4	15	11.31
RM 调度算法	3	14	10.68

5.2 截止期错失率

截止期错失率 (Deadline Miss Rate, DMR) 表示在某段时间内系统中未满足截止期的实时任务数量与实时任务总数之比,是衡量实时性能的另一个重要指标. 截止期错失率越低,实时性能越好.

图 3 为三种算法在相同条件下截止期错失率随实时任务数量增加的变化情况,其中横轴表示实时任务数量,纵轴表示截止期错失率. 从图中可以看出,本文算法的截止期错失率随负荷的增加,增长速度比 EDF 调度算法稍快,而远慢于 RM 调度算法. 说明本文算法的截止期错失率性能介于 EDF 调度算法与 RM 调度算法之间.

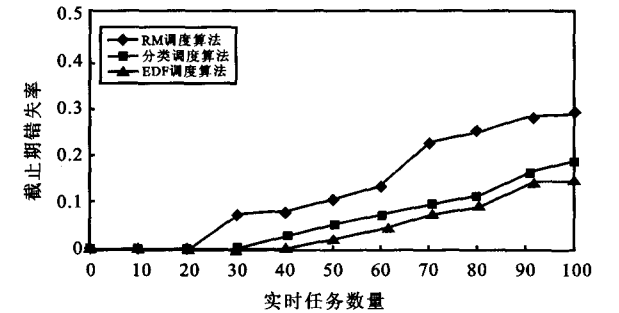


图 3 截止期错失率测试结果对比

6 结束语

本文提出了一种分类调度算法,使用两种改进的调度算法,并规定不同类型的实时任务可以灵活选择适合的调度算法进行调度. 另外,本文算法的可调度性分析简单,使用其中任何一种调度算法的实时任务的可调度性分析独立于使用其中另一种调度算法的实时任务. 以调度延迟和截止期错失率为性能指标,对包含 100 个任务的混合实时任务集用本文算法与 EDF 和 RM 调度算法作对比. 实验结果表明,本文算法具有 EDF 和 RM 调度算法各自的优点,综合实时性能有了很大程度的提高.

参考文献:

[1] Jin Hong, Wang Hongan, Wang Qiang. A real-time scheduling algorithm based on priority table and its implementation[J]. Journal of Software, 2004,15(3): 360-370.

[2] 刘胜辉, 马嵩. 基于 Linux 内核的实时调度机制研究及应用[J]. 计算机工程与应用, 2008,44(6): 121-123.

[3] 萧伟, 冯治宝, 应启蔓. 改进型 EDF 调度算法的研究与实现[J]. 计算机工程, 2009,35(18): 231-233.

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

[5] 董吉文, 张阳. 嵌入式实时操作系统任务调度算法的改进与应用[J]. 计算机应用, 2009,29(9):2516-2519.

[6] 谭云福, 刘杰, 刘国华. Linux 中一种改进的实时调度算法及其应用[J]. 计算机科学, 2008,35(10):256-258.

[7] Jin Hong, Wang Hongan, Wang Qiang. An integrated design method of task priority[J]. Journal of Software, 2003,14(3):376-382.

[8] Dana M, Pascale M, Laurent G. Analysis of deadline assignment methods in distributed real-time systems[J]. Computer Communications, 2004,27(15):1412-1423.

[9] Terrasa A, Garcia Fomes A, Botti V J. Flexible real-time linux: a flexible hard real-time environment[J]. Real-Time Systems, 2004,22(2):151-173.

作者简介:

霍华 男,(1968-),博士后,硕士生导师,副教授. 研究方向为智能信息处理、光纤通道技术、嵌入式系统.

崔欢欢 女,(1985-),硕士研究生. 研究方向为嵌入式系统.