

一种基于分组的多核嵌入式实时调度算法

康 鹏,刘从新,沈绪榜

(西安微电子技术研究所,陕西 西安 710054)

摘 要: 针对亲和多核调度方法中容易出现负载不均衡问题,在原有亲和多核调度算法中,提出了一种新的调度算法.该算法在每个核上增加统计任务对 CPU 进行实时监控,同时在系统任务中增加全局负载均衡任务进行第三级调度,相对于已有的亲和调度算法,在保证原有操作系统不被破坏的基础上,操作系统核间负载更加均衡.通过组内亲和和任务融合,来减小系统功耗,满足了操作系统中任务调度的实时性及负载均衡的需要.

关键词: 多核调度; VxWorks; 亲和调度; 组亲和

中图分类号: TP316.2

文献标识码: A

文章编号: 1000-7180(2016)10-0032-04

Multicore Embedded Real-Time Scheduling Algorithm Based on Gang Scheduling

KANG Peng, LIU Cong-Xin, SHEN Xu-Bang

(Xi'an Microelectronics Technology Institute, Xi'an 710054, China)

Abstract: Aimed at solving the unbalanced problem caused by the Real-time task scheduling on multi-core, this paper proposed an advanced scheduling based on affinity scheduling. This algorithm monitors CPU by increasing a statistical task. Meanwhile, it can process the third scheduling by adding global payload task. Compared with the former scheduling, this method guarantees the balance of every CPU and the operation system undamaged. Real-time task scheduling and payload balance can be satisfied by mixing the different CPU unites and reducing the system power consumption.

Key words: task scheduling on multi-core; VxWorks; affinity scheduling; gang affinity

1 引言

在并发或并行程序设计,经常会有多个进程或者线程同时竞争 CPU. 不管任何时刻,当这些进程或者线程同时处于就绪态,竞争便会发生. 如果只有一个 CPU 可用,操作系统需要做出一个选择:接下来运行哪一个进程. 做选择的这部分操作系统被称为调度器,它所使用的算法称为调度算法^[1]. 在嵌入式系统中,调度算法一般指的是线程调度. 在多核系统中,调度器需要决定:运行哪一个线程以及线程将在哪一个 CPU 上运行^[2]. 本文通过研究 VxWorks 嵌入式实时操作系统的调度算法,针对多核操作系统中任务调度存在的问题,提出了组亲和算法,以解决亲和调度算法中负载不均衡问题,同时减

小系统功耗.

2 通用的多核调度算法

通常在线程调度中会遇到两种情形,线程相关和线程不相关. 本文主要讨论线程不相关的多核调度算法.

对于不相关线程,最简单的调度算法是构建一个全局链表集合来对就绪线程进行管理,每个链表具有不同的优先级,同链表下的线程优先级相同. 例如,图 1(a)中有 8 个处理器正在并行处理,此时一个有着 8 个线程的优先级集合正在等待运行. 第一个完成当前工作的 CPU 是 4 号,然后 4 号 CPU 锁定调度队列和选择最高优先级线程 A,如图 1(b)所示. 接下来,2 号 CPU 开始空闲,调度器选择线程 B

来使用 2 号 CPU,如图 1(c)所示. 只要线程完全不相关,使用这种方法的调度是一个合理的选择,同时实现起来也是最简单高效的.

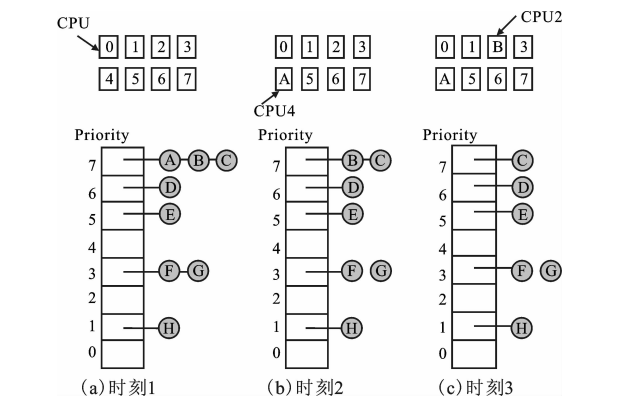


图 1 使用单一调度数据结构的链表集合进行多核任务调度

在多核处理器中,操作系统使用全局链表来对任务进行管理时,算法自动对整个系统提供了负载均衡,链表的维护也更简单.一旦发现有空闲核可以使用,操作系统便调度高优先级线程使用这个核,因而不会有核处于空闲状态.这种方法有两方面的缺点:(1)调度数据结构是核间共享的,每次只能有一个核可以使用,随着处理器核数的增多,全局链表的访问冲突更加频繁.(2)当线程被阻塞,核间上下文切换比较耗时.因为核间线程切换需要重新加载该线程的 Cache,所以频繁的核间切换将大幅度降低系统性能.

针对上述缺点,一些多核操作系统采用了一种称为亲和调度(Affinity Scheduling)的算法.这种算法的基本思想是对于即将使用 CPU 的线程,让其尽量继续运行在上次运行的 CPU 上.亲和调度的实现过程为:当创建一个线程时,由用户层来决定该线程在哪个 CPU 上运行.通过这个策略,每个 CPU 获得它自己的线程集合^[3].

线程调度由核内调度算法实现.每一个 CPU 使用优先级或者时间片轮转来对自己核上的任务进行调度.通过保持一个线程一直运行在一个 CPU 上,从而减小因为线程核间切换造成的 cache 不命中.但是,如果一个 CPU 没有线程来运行,它将取从另一个 CPU 上获得一个线程而不是一直保持空闲状态.

该算法调度有如下优点.首先,负载被平均地分配到可用的 CPU 上.其次,尽可能地减小了 cache 不命中.最后,通过为每一个 CPU 分配就绪任务链表,极大地减小了就绪任务链表的冲突,因为尝试使用其他 CPU 的就绪任务链表是不会发生的.

3 VxWorks 多核任务调度

VxWorks 内核的多核任务调度机制是底层基于优先级的抢占式调度,上层使用亲和调度算法.在 VxWorks 内核中,任务的就绪队列有两种:一种是全局就绪队列(所有 CPU 都可共享),另一种是为每个 CPU 分配一个独享的亲和就绪队列.当有 N 个 CPU 时,共有 $N+1$ 个就绪队列^[4].任务调度的决策能在很短的时间做出,即哪个任务在哪个 CPU 上执行.因为每个 CPU 只能执行针对自己的亲和就绪队列和全局就绪队列上的任务,无法执行其他 CPU 亲和就绪队列上的任务.此外,调度器可以保留一些 CPU 让亲和任务独享,不允许全局任务抢占该保留的 CPU.这样做可以提升该保留 CPU 上亲和队列任务集的性能.例如,这样可以使负载更重的网络相关任务更好的处理速度.

图 2 是 VxWorks 多核操作系统内核的任务状态转换图,使用任务所在队列的调度关系,可以将图中的任务状态转换的调度规则划分为两种类型:任务离开就绪队列和任务进入就绪队列.

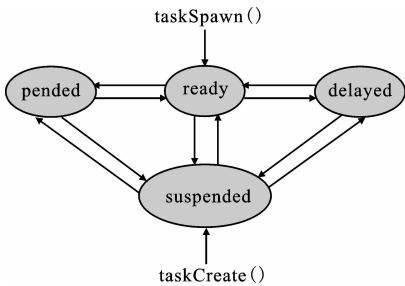


图 2 任务的基本的状态转换图

对于任务离开就绪队列,即任务状态从就绪态转换为其他三种状态之一.任务的调度规则如下:如果任务属于全局就绪队列,将任务从全局队列中移除.如果任务属于亲和队列,将任务从亲和队列中移除.选择一个新任务到 CPU 中运行,选择的新的任务只能是指定 CPU 的亲和队列的第一个任务,或者是比这个亲和任务优先级更高的、当前没有运行或没有被调度运行的全局任务.

另一种情况,对于任务进入就绪队列,即任务状态从其他三种状态之一转换为就绪态.任务的调度规则分为两种情形:(1)任务是亲和 CPU 的任务.任务进入亲和就绪队列.如果新任务比当前 CPU 上正在执行任务的优先级高,进行任务切换,发出重调度请求.假如被切换出去的任务位于全局就绪队列,则任务调度器会在上下文切换前,为

切换出去的全局任务寻找合适的 CPU 使用, 如果找到正在运行更低优先级任务的 CPU, 则通知找到的 CPU 进行任务调度. (2) 任务属于全局就绪队列. 将任务加入全局就绪队列中. 在所有当前正在运行任务的 CPU 中, 寻找优先级最低的任务. 如果找到, 则向运行这个优先级最低任务的 CPU 发出重调度请求.

例如图 3(a) 中, 4 个 CPU 上分别运行着 4 个任务, CPU 3 运行着全局就绪队列上的任务. 在某时刻, 任务 E 被阻塞挂起, 此时 CPU 3 上进行任务调度, 将任务 E 从全局队列中移出, 由于任务 F 的优先级高于任务 D, 则将任务 F 获得 CPU 3 的使用权, 如图 3(b) 所示. 接下来, 任务 H 被唤醒, 加入到亲和就绪队列 3 中, 此时, 由于任务 H 的优先级高于任务 F, 则任务 F 让出 CPU 3 的使用权, 并记录等待二次调度. 任务 H 获得 CPU 3 的使用权后, 进行二次调度, 发现当前正在运行的最低优先级的任务是任务 B, 运行在 CPU 1 上, 于是发送调度请求, 通知 CPU 1 进行任务调度. CPU 1 在执行过程中发现需要进行任务调度, 检查全局就绪队列和自己的亲和就绪队列, 发现任务 F 的优先级比当前任务 B 的优先级高, 于是进行任务切换, 任务 F 获得操作系统的使用权, 如图 3(c) 所示.

上述方法虽然可以解决全局链表冲突的问题, 但是容易出现负载不均衡的问题, 即一些核上任务繁重, 另一些核可能一直处于空闲状态. 针对上述问题, 本文提出了如下改进算法.

4 组亲和调度算法

之前的算法中, 每个亲和任务只能亲和一个 CPU.

本文提出了一种改进的亲和算法, 即组亲和算法. 在该算法中一个任务可以亲和多个 CPU, 这里为了简化算法实现, 一个任务可以亲和两个 CPU, 即亲和偶数号 CPU 的任务可以亲和该 CPU 相邻的奇数 CPU.

为了实现该算法, 增加了 $N+1$ 个任务.

(1) 每个 CPU 核上增加了一个统计任务, 统计任务的优先级为最低优先级减一, 主要工作是统计当前 CPU 的使用率, 供负载均衡任务使用. 同时, 统计任务如果发现当前 CPU 的使用率过高, 则唤醒负载均衡任务.

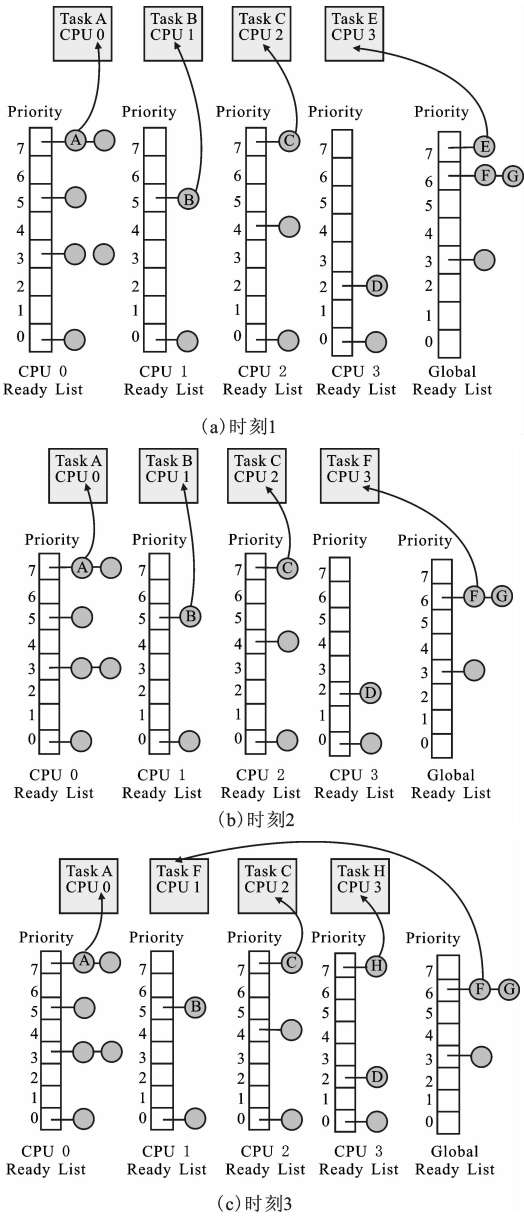


图 3 多核任务调度示例

(2) 在全局任务队列中新增一个负载均衡任务, 负载均衡任务的优先级为最高优先级加一, 防止与系统任务平分 CPU 时间, 影响系统响应时间.

负载均衡任务的主要工作是: (1) 依次检查每个 CPU 的使用率是否超过系统高负载警戒值, 如果超过这个值, 则该任务调度就绪队列上的次优先级任务到组亲和的另一个 CPU 上, 并修改该任务的亲和号. 如果另一个 CPU 处于关闭状态, 则打开 CPU, 进行任务分离. (2) 该任务还会检查此时 CPU 的使用率是否低于节能线. 如果低于节能线, 则检查该 CPU 的使用率加上该组另一个 CPU 的使用率是否小于负载保护线 (负载保护线不能超过 70%, CPU 的使用率一旦超过临界线将无法保证任务的

正常调度),如果小于负载保护线,说明可以进行节能管理,则通知负载均衡任务将使用率较低的 CPU 上的任务分配到该组的另一个 CPU 上,并关闭该 CPU.当任务在唤醒后,根据亲和性检查当前 CPU 是否处于关闭状态,如果处于关闭状态,则选择组亲和的另一个 CPU.

该算法的实现过程如图 4 所示:假设 CPU1 在启动后,CPU 的使用率位于负载保护线和节能线之间,经过一段时间后,CPU1 的使用率达到高负载警戒线,此时在 CPU0 上运行的负载均衡任务发现 CPU1 的使用率超过了负载保护线,则负载均衡任务将 CPU1 上的次优先级任务调度到 CPU0 上运行,并减小 CPU1 的使用率.在每次执行完一次任务调度后,负载均衡任务挂起自己,等待由统计任务发出的信号量.

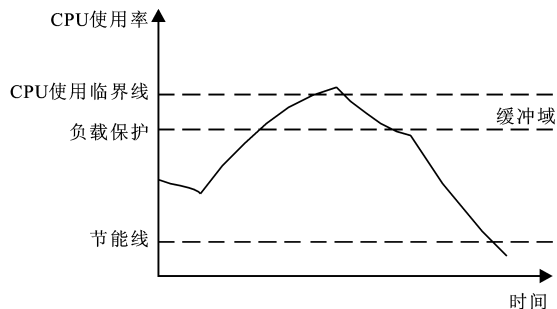


图 4 CPU 负载变化图

为了保证实时性,可以预留前两个 CPU 不参与组亲和算法,这两个 CPU 上的任务仅仅亲和对应的 CPU.可以将一些重要的高优先级任务放到这两个 CPU 中,从而及时处理中断.因为负载均衡任务对 CPU 的使用率调度是需要时间过渡的,可能导致一些重要的高优先级任务得不到及时处理.

通过上述机制,在确保实时性的基础上,实现组亲和调度算法,解决亲和调度算法中负载均衡的问题,以及在处理器利用率较低时,实现组内亲和和队列的融合,关闭某些 CPU,达到减小系统功耗的

目的.

该算法也会带来一些问题,由于每个核增加了一个统计任务,可能会增加每个核的系统负载.

5 结束语

本文针对 VxWorks 嵌入式实时多核操作系统现有亲和任务调度方法存在的多核负载不均衡问题,提出了改进的组亲和调度算法,与原来的亲和调度算法相比,理论上该方法在不破坏实时性的前提下,解决了亲和调度算法中 CPU 负载不均衡的问题,同时降低了系统功耗.后继实验验证将在下一步的工作中进行,同时也将对组亲和调度算法的分组粒度开展更加深入的研究.

参考文献:

- [1] Vaswani R, Zahorjan J. The implications of cache affinity on processor scheduling for multiprogrammed shared-memory multiprocessors [C] // Proc. 13th Symp. On Operation systems Principle. USA. Pacific Grove, ACM, 1991:26-40.
- [2] Andrew S Tanenbaum. 现代操作系统[M]. 3 版. 北京:机械工业出版社,2009.
- [3] Jean J Labrosse. 嵌入式实时操作系统 uC/OS-II[M]. 2 版. 北京:北京航空航天大学出版社,2003.
- [4] John L Hennessy, David A Patterson. 计算机体系结构:量化研究方法[M]. 5 版. 北京:机械工业出版社,2011.

作者简介:

康 鹏 男,(1991-),硕士研究生.研究方向为嵌入式操作系统. E-mail:734289208@qq.com.

刘从新 男,(1976-),博士,高级工程师.研究方向为多核并行框架.

沈绪榜 男,(1931-),研究员,博士生导师.中国科学院院士.研究方向为嵌入式计算机及其芯片.

(上接第 31 页)

- [5] Goldstein S C, Schmit H PipeRench. A reconfigurable architecture and compiler[C]// Proceedings of IEEE International Symposium on Circuits and Systems. New York, 2000: 70-77.
- [6] 肖钰,刘雷波. 应用于视频处理器的可重构流水处理器的设计与实现[J]. 电视技术,2008,32(6):18-20.
- [7] 沈绪榜,刘泽响,王茹. 计算机体系结构的统一模型[J]. 计算机学报,2007,29(5):729-736.
- [8] 牟胜梅,李兆刚. 一种面向 FPGA 的指/对数函数求

值算法[J]. 计算机工程与应用,2011,33(4):59-61.

作者简介:

吕 青 女,(1989-),硕士研究生.研究方向为集成电路系统设计. E-mail:390396308@qq.com

蒋 林 男,(1970-),教授.研究方向为专用集成电路设计.

邓军勇 男,(1981-),副教授.研究方向为专用集成电路设计.