



(12) 发明专利

(10) 授权公告号 CN 113687909 B

(45) 授权公告日 2024. 01. 30

(21) 申请号 202110859497.2

CN 102253857 A,2011.11.23

(22) 申请日 2021.07.28

CN 102662763 A,2012.09.12

(65) 同一申请的已公布的文献号

CN 103678003 A,2014.03.26

申请公布号 CN 113687909 A

CN 108037994 A,2018.05.15

(43) 申请公布日 2021.11.23

CN 108459906 A,2018.08.28

(73) 专利权人 华东计算技术研究所(中国电子科技集团公司第三十二研究所)

CN 110321212 A,2019.10.11

CN 110347507 A,2019.10.18

CN 110968876 A,2020.04.07

地址 201800 上海市嘉定区嘉罗路1485号

CN 112817690 A,2021.05.18

US 2011119422 A1,2011.05.19

(72) 发明人 左龙 韩笑 汪云云

US 2011154321 A1,2011.06.23

(74) 专利代理机构 上海段和段律师事务所
31334

Taehyung Lee,et al.,."VCPU

专利代理师 李佳俊 郭国中

Prioritization Interface for Improving

the Performance of Latency-Critical

Tasks".《2020 14th International

Conference on Ubiquitous Information

Management and Communication (IMCOM)》

.2020,全文.

(续)

(51) Int. Cl.

G06F 9/455 (2006.01)

G06F 9/50 (2006.01)

G06F 9/48 (2006.01)

(56) 对比文件

审查员 李桂红

CN 101169731 A,2008.04.30

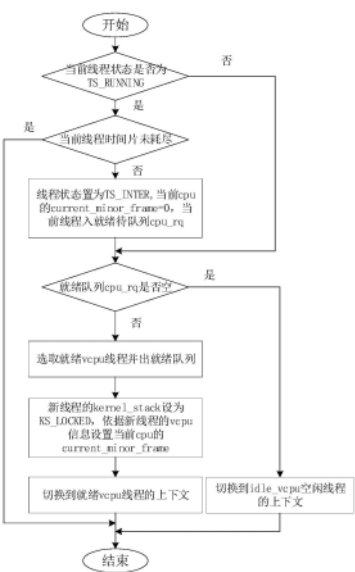
权利要求书4页 说明书8页 附图2页

(54) 发明名称

基于微内核的分时vcpu多核调度方法及系统

(57) 摘要

本发明提供一种基于微内核的分时vcpu多核调度方法及系统,涉及虚拟化技术领域,该方法包括:调度初始化步骤:将物理处理器核心pcpu进行分组;每个pcpu申请内存空间创建各自的就绪队列、等待队列,并为每个pcpu创建各自的空闲idle线程;分区多核调度步骤:在vcpu线程调度时,判断当前vcpu线程的时间片是否耗尽;就绪线程选择步骤:在分区多核调度步骤中,选择就绪线程,即vcpu线程或普通任务线程时,对比判断分区处于新周期还是旧周期,选择相应的就绪线程进行切换。本发明能够基于微内核实现,具有更高的灵活性和可靠性;分时调度算法,有效整合实时和非实时任务所需的资源;能够实时分区调度算法提高该分区任务调度的实时性。



CN 113687909 B

[接上页]

(56) 对比文件

戴志华. Xen上的CPU性能监视系统. 舰船电子工程. 2012, (第05期), 全文.

李亮; 吴庆波. Linux超线程感知的调度算法研究. 电子技术应用. 2006, (第02期), 全文.

邓昀; 程小辉; 王新政. 微内核结构嵌入式实时操作系统的研究与设计. 微电子学与计算机

. 2012, (第10期), 全文.

冯华; 卢凯; 王小平. 面向多核处理器的实时优化技术: 基于独立实时域的实时优化方法. 计算机科学. 2013, (第09期), 全文.

周霆; 胡宁; 任晓瑞. 一种分区操作系统实时任务调度方法. 计算机光盘软件与应用. 2013, (第10期), 全文.

1. 一种基于微内核的分时vcpu多核调度方法,其特征在于,包括:

调度初始化步骤:将物理处理器核心pcpu进行分组,一组注册为实时调度策略作为实时分区;另一组注册为非实时调度策略作为普通分区,每组为一个分区共享同种调度策略,分区间互不干扰;每个pcpu申请内存空间创建各自的就绪队列、等待队列,并为每个pcpu创建各自的空闲idle线程;

分区多核调度步骤:在vcpu线程调度时,判断当前vcpu线程或普通任务线程的时间片是否耗尽,若未耗尽时间片则不能打断;

若耗尽时间片,则选取当前pcpu的就绪队列中首个就绪线程,即vcpu线程或普通任务线程,进行切换,若当前pcpu的就绪队列为空则直接运行空闲idle线程;

就绪线程选择步骤:在分区多核调度步骤中,选择就绪线程,即vcpu线程或普通任务线程时,通过当前时间与截止时间的对比判断分区处于新周期还是旧周期,并判断是否启动了虚拟机,根据判断结果选择相应的就绪线程进行切换;

其中,若处于旧周期且启动了虚拟机,即就绪线程为vcpu线程而非普通任务线程,则直接选择当前pcpu就绪队列中首个就绪的vcpu线程进行切换;

若处于新周期且启动了虚拟机,即就绪线程为vcpu线程而非普通任务线程,则计算当前新周期的截止时间并切换至当前pcpu就绪队列中的首个就绪的vcpu线程;

若处于新周期但没有启动虚拟机,即就绪线程为普通任务线程而非vcpu线程,则需要重新选择本分区内有效的pcpu并切换至该pcpu的就绪队列中首个就绪的普通任务线程,当切换就绪线程而就绪队列为空时则切换至当前pcpu的空闲idle线程。

2. 根据权利要求1所述的基于微内核的分时vcpu多核调度方法,其特征在于,所述调度初始化步骤包括:

步骤S1.1:初始化开始;

步骤S1.2:将物理处理器核心pcpu进行分组,一组注册为实时调度策略作为实时分区;另一组注册为非实时调度策略作为普通分区;

步骤S1.3:为每个pcpu申请内存空间创建各自的就绪队列和等待队列,并将就绪队列和等待队列地址记录在全局数组cpu_rq[]与cpu_wq[]中;

步骤S1.4:为每个pcpu创建各自的空闲idle线程,并将该idle线程标识记录在全局数组cpu_it[]中;

步骤S1.5:pcpu开始运行各自的空闲idle线程;

步骤S1.6:进行首次调度,当就绪队列不空时,pcpu切换到就绪队列中首个就绪线程运行,即vcpu线程或普通任务线程,否则仍然运行空闲idle线程;

步骤S1.7:结束并返回。

3. 根据权利要求1所述的基于微内核的分时vcpu多核调度方法,其特征在于,所述分区多核调度步骤具体包括:

步骤S2.1:非首次调度开始,若此时pcpu上正在运行空闲idle线程则切换回空闲idle线程后跳转步骤S2.9,若此时正在运行vcpu线程或普通任务线程则跳转步骤S2.2;

步骤S2.2:判断当前运行的线程的状态,thread_state是否为正在运行TS_RUNNING,若为TS_RUNNING则跳转步骤S2.3,若不为TS_RUNNING则跳转步骤S2.4;

步骤S2.3:判断当前运行的线程的时间片是否耗尽,若耗尽则跳转步骤S2.4,若未耗尽

则无需调度直接跳转步骤S2.9;

步骤S2.4:设置当前线程状态thread_state为TS_INTER,设置当前线程时间片current_minor_frame为0,依据当前线程与物理处理器核pcpu的亲和性affinity加入相应的pcpu的就绪队列中;

步骤S2.5:判断当前pcpu的就绪队列是否为空,若为空则当前pcpu切换到空闲idle线程的上下文运行后,跳转步骤S2.9,不为空则跳转步骤S2.6;

步骤S2.6:在当前pcpu的就绪队列中选取就绪线程,即vcpu线程或普通任务线程,并将其出队;

步骤S2.7:设置KS_LOCKED标志位锁住就绪线程的栈kernel_stack,依据该线程信息设置该线程的时间片current_minor_frame;

步骤S2.8:解锁线程的kernel_stack,切换到该线程的上下文运行;

步骤S2.9:结束。

4.根据权利要求1所述的基于微内核的分时vcpu多核调度方法,其特征在于,所述就绪线程选择步骤具体包括:

步骤S3.1:开始;

步骤S3.2:比较当前时间与当前pcpu运行线程,即vcpu线程或普通任务线程的截止时间,判断当前线程是否处于新周期中,若是则跳转步骤S3.3,若否则跳转步骤S3.11;

步骤S3.3:判断当前是否运行了虚拟机,若运行了虚拟机则当前运行的线程为vcpu线程跳转步骤S3.4,若未运行虚拟机则当前运行线程为普通任务线程跳转步骤S3.5;

步骤S3.4:在多个物理处理器核pcpu中寻找有效的pcpu并记录其索引;

步骤S3.5:记录此次切换的当前时间作为next_switch_time,和新周期的截止时间作为next_major_frame;

步骤S3.6:判断当前pcpu的就绪队列,即vcpu线程或普通任务线程是否为空,不为空则跳转步骤S3.7,空则跳转步骤S3.14;

步骤S3.7:判断当前pcpu的就绪队列中首个就绪线程的栈kernel_stack的状态是否为空闲KS_FREE,空闲则跳转步骤S3.8,非空闲则跳转步骤S3.14;

步骤S3.8:将当前pcpu就绪队列中首个就绪线程出队;

步骤S3.9:判断出队的就绪线程状态是否为结束退出TE_EXITING,是则跳转步骤S3.7,否则跳转步骤S3.10;

步骤S3.10:切换到该就绪线程的上下文运行,跳转步骤S3.15结束并返回该线程号;

步骤S3.11:比较当前时间与当前pcpu运行线程的截止时间,判断当前线程是否处于旧周期中,若是则跳转步骤S3.12,若否则跳转步骤S3.14;

步骤S3.12:判断当前是否运行了虚拟机,若运行了虚拟机则当前运行的线程为vcpu线程跳转步骤S3.6,若未运行虚拟机则当前运行线程为普通任务线程跳转步骤S3.14;

步骤S3.13:记录此次切换的当前时间作为next_switch_time,跳转步骤S3.7;

步骤S3.14:切换到空闲idle线程;

步骤S3.15:结束并返回该线程号。

5.一种基于微内核的分时vcpu多核调度系统,其特征在于,包括:

调度初始化模块:将物理处理器核心pcpu进行分组,一组注册为实时调度策略作为实

时分区;另一组注册为非实时调度策略作为普通分区,每组为一个分区共享同种调度策略,分区间互不干扰;每个pcpu申请内存空间创建各自的就绪队列、等待队列,并为每个pcpu创建各自的空闲idle线程;

分区多核调度模块:在线程调度时,判断当前线程,即vcpu线程或普通任务线程,的时间片是否耗尽,若未耗尽时间片则不能打断;

若耗尽时间片,则选取当前pcpu的就绪队列中首个就绪线程,即vcpu线程或普通任务线程进行切换,若当前pcpu的就绪队列为空则直接运行空闲idle线程;

就绪线程选择模块:在分区多核调度模块中,选择就绪线程,即vcpu线程或普通任务线程时,通过当前时间与截止时间的对比判断分区处于新周期还是旧周期,根据判断结果选择就绪线程进行切换;

其中,若处于旧周期且启动了虚拟机,即就绪线程为vcpu线程而非普通任务线程,则直接选择当前pcpu就绪队列中首个就绪的vcpu线程进行切换;

若处于新周期且启动了虚拟机,即就绪线程为vcpu线程而非普通任务线程,则计算当前新周期的截止时间并切换至当前pcpu就绪队列中的首个就绪的vcpu线程;

若处于新周期但没有启动虚拟机,即就绪线程为普通任务线程而非vcpu线程,则需要重新选择本分区内有效的pcpu并切换至该pcpu的就绪队列中首个就绪的普通任务线程,当切换就绪线程而就绪队列为空时则切换至当前pcpu的空闲idle线程。

6.根据权利要求5所述的基于微内核的分时vcpu多核调度系统,其特征在于,所述调度初始化模块包括:

模块M1.1:初始化开始;

模块M1.2:将物理处理器核心pcpu进行分组,一组注册为实时调度策略作为实时分区;另一组注册为非实时调度策略作为普通分区;

模块M1.3:为每个pcpu申请内存空间创建各自的就绪队列和等待队列,并将就绪队列和等待队列地址记录在全局数组cpu_rq[]与cpu_wq[]中;

模块M1.4:为每个pcpu创建各自的空闲idle线程,并将该idle线程标识记录在全局数组cpu_it[]中;

模块M1.5:pcpu开始运行各自的空闲idle线程;

模块M1.6:进行首次调度,当就绪队列不空时,pcpu切换到就绪队列中首个就绪线程,即vcpu线程或普通任务线程运行,否则仍然运行空闲idle线程;

步骤M1.7:结束并返回。

7.根据权利要求5所述的基于微内核的分时vcpu多核调度系统,其特征在于,所述分区多核调度模块具体包括:

模块M2.1:非首次调度开始,若此时pcpu上正在运行空闲idle线程则切换回空闲idle线程后跳转模块M2.9,若此时正在运行vcpu线程或普通任务线程则跳转模块M2.2;

模块M2.2:判断当前运行的线程的状态,thread_state是否为正在运行TS_RUNNING,若为TS_RUNNING则跳转模块M2.3,若不为TS_RUNNING则跳转模块M2.4;

模块M2.3:判断当前运行的线程的时间片是否耗尽,若耗尽则跳转模块M2.4,若未耗尽则无需调度直接跳转模块M2.9;

模块M2.4:设置当前线程状态thread_state为TS_INTER,设置当前线程时间片

current_minor_frame为0,依据当前线程与物理处理器核pcpu的亲和性affinity加入相应的pcpu的就绪队列中;

模块M2.5:判断当前pcpu的就绪队列是否为空,若为空则当前pcpu切换到空闲idle线程的上下文运行后,跳转模块M2.9,不为空则跳转模块M2.6;

模块M2.6:在当前pcpu的就绪队列中选取就绪线程,即vcpu线程或普通任务线程,并将其出队;

模块M2.7:设置KS_LOCKED标志位锁住就绪线程的栈kernel_stack,依据该线程信息设置该线程的时间片current_minor_frame;

模块M2.8:解锁线程的kernel_stack,切换到该线程的上下文运行;

模块M2.9:结束。

8.根据权利要求5所述的基于微内核的分时vcpu多核调度系统,其特征在于,所述就绪线程选择模块具体包括:

模块M3.1:开始;

模块M3.2:比较当前时间与当前pcpu运行线程的截止时间,即vcpu线程或普通任务线程,判断当前线程是否处于新周期中,若是则跳转模块M3.3,若否则跳转模块M3.11;

模块M3.3:判断当前是否运行了虚拟机,若运行了虚拟机则当前运行的线程为vcpu线程跳转模块M3.4,若未运行虚拟机则当前运行线程为普通任务线程跳转模块M3.5;

模块M3.4:在多个物理处理器核pcpu中寻找有效的pcpu并记录其索引;

模块M3.5:记录此次切换的当前时间作为next_switch_time,和新周期的截止时间作为next_major_frame;

模块M3.6:判断当前pcpu的就绪队列,即vcpu线程或普通任务线程是否为空,不为空则跳转模块M3.7,空则跳转模块M3.14;

模块M3.7:判断当前pcpu的就绪队列中首个就绪线程的栈kernel_stack的状态是否为空闲KS_FREE,空闲则跳转模块M3.8,非空闲则跳转模块M3.14;

模块M3.8:将当前pcpu就绪队列中首个就绪线程出队;

模块M3.9:判断出队的就绪线程状态是否为结束退出TE_EXITING,是则跳转模块M3.7,否则跳转模块M3.10;

模块M3.10:切换到该就绪线程的上下文运行,跳转模块M3.15结束并返回该线程号;

模块M3.11:比较当前时间与当前pcpu运行线程的截止时间,判断当前线程是否处于旧周期中,若是则跳转模块M3.12,若否则跳转模块M3.14;

模块M3.12:判断当前是否运行了虚拟机,若运行了虚拟机则当前运行的线程为vcpu线程跳转模块M3.6,若未运行虚拟机则当前运行线程为普通任务线程跳转模块M3.14;

模块M3.13:记录此次切换的当前时间作为next_switch_time,跳转模块M3.7;

模块M3.14:切换到空闲idle线程;

模块M3.15:结束并返回该线程号。

基于微内核的分时vcpu多核调度方法及系统

技术领域

[0001] 本发明涉及虚拟化技术领域,具体地,涉及一种基于微内核的分时vcpu多核调度方法及系统。

背景技术

[0002] 伴随嵌入式领域的不断发展,嵌入式系统的应用场合越来越多,面对日益增多的用户需求,只能支持单一应用类型的嵌入式系统已经无法满足要求。为了应对这一多样化的挑战,现有的嵌入式系统往往包含多个子系统,多种子系统满足多种需求,如实时性和非实时性需求,低安全等级和高安全等级需求等。多样化的子系统也增加了嵌入式系统的设计复杂度。在此背景下,嵌入式行业需要新的解决方案以实现多种子系统的资源整合,虚拟化技术在嵌入式系统的应用为此提供了有力的支撑。嵌入式系统的虚拟化技术能够在同一硬件平台上虚拟出多个分区,不同分区均可运行独立的操作系统或应用,这使得实时操作系统和通用操作系统并存,方便了对实时和非实时应用的整合。

[0003] vcpu即电脑中的虚拟处理器,相对于物理CPU而言,虚拟机内的CPU。

[0004] 公开号为CN112579294A的发明专利,公开了一种虚拟机的多核调度的实现方法及装置,所述方法包括:将一虚拟机的主虚拟机与N-1个影子虚拟机对应配置到N个CPU核上;通过一时间调度表,调度所述主虚拟机和所述影子虚拟机在相同的时间窗口下分别运行于对应的CPU核上;所述时间调度表配置有用于调度主虚拟机和影子虚拟机运行的时间窗口。

[0005] 然而,现有的开源虚拟化技术如基于Linux的KVM,大多基于宏内核,这使得分区间、任务间的隔离性和可靠性有所下降;而且这些开源虚拟化技术,大多只具备软实时性能,无法满足强实时应用场景的要求。

发明内容

[0006] 针对现有技术中的缺陷,本发明提供一种基于微内核的分时vcpu多核调度方法及系统。

[0007] 根据本发明提供的一种基于微内核的分时vcpu多核调度方法及系统,所述方案如下:

[0008] 第一方面,提供了一种基于微内核的分时vcpu多核调度方法,所述方法包括:

[0009] 调度初始化步骤:将物理处理器核心pcpu进行分组;每个pcpu申请内存空间创建各自的就绪队列、等待队列,并为每个pcpu创建各自的空闲idle线程;

[0010] 分区多核调度步骤:在vcpu线程调度时,判断当前vcpu线程或普通任务线程的时间片是否耗尽,若未耗尽时间片则不能打断;

[0011] 若耗尽时间片,则选取当前pcpu的就绪队列中首个就绪线程,即vcpu线程或普通任务线程,进行切换,若当前pcpu的就绪队列为空则直接运行空闲idle线程;

[0012] 就绪线程选择步骤:在分区多核调度步骤中,选择就绪线程,即vcpu线程或普通任务线程时,通过当前时间与截止时间的对比判断分区处于新周期还是旧周期,并判断是否

启动了虚拟机,根据判断结果选择相应的就绪线程进行切换。

[0013] 优选的,所述调度初始化步骤包括:

[0014] 步骤S1.1:初始化开始;

[0015] 步骤S1.2:将物理处理器核心pcpu进行分组,一组注册为实时调度策略作为实时分区;另一组注册为非实时调度策略作为普通分区;

[0016] 步骤S1.3:为每个pcpu申请内存空间创建各自的就绪队列和等待队列,并将就绪队列和等待队列地址记录在全局数组cpu_rq[]与cpu_wq[]中;

[0017] 步骤S1.4:为每个pcpu创建各自的空闲idle线程,并将该idle线程标识记录在全局数组cpu_it[]中;

[0018] 步骤S1.5:pcpu开始运行各自的空闲idle线程;

[0019] 步骤S1.6:进行首次调度,当就绪队列不空时,pcpu切换到就绪队列中首个就绪线程运行,即vcpu线程或普通任务线程,否则仍然运行空闲idle线程;

[0020] 步骤S1.7:结束并返回。

[0021] 优选的,所述分区多核调度步骤具体包括:

[0022] 步骤S2.1:非首次调度开始,若此时pcpu上正在运行空闲idle线程则切换回空闲idle线程后跳转步骤S2.9结束并返回idle线程,若此时正在运行vcpu线程或普通任务线程则跳转步骤S2.2;

[0023] 步骤S2.2:判断当前运行的线程的状态,thread_state是否为正在运行TS_RUNNING,若为TS_RUNNING则跳转步骤S2.3,若不为TS_RUNNING则跳转步骤S2.4;

[0024] 步骤S2.3:判断当前运行的线程的时间片是否耗尽,若耗尽则跳转步骤S2.4,若未耗尽则无需调度直接跳转步骤S2.9结束并返回;

[0025] 步骤S2.4:设置当前线程状态thread_state为TS_INTER,设置当前线程时间片current_minor_frame为0,依据当前线程与物理处理器核pcpu的亲亲和性affinity加入相应的pcpu的就绪队列中;

[0026] 步骤S2.5:判断当前pcpu的就绪队列是否为空,若为空则当前pcpu切换到到空闲idle线程的上下文运行后,跳转步骤S2.9结束并返回,不为空则跳转步骤S2.6;

[0027] 步骤S2.6:在当前pcpu的就绪队列中选取就绪线程,即vcpu线程或普通任务线程,并将其出队;

[0028] 步骤S2.7:设置KS_LOCKED标志位锁住就绪线程的栈kernel_stack,依据该线程信息设置该线程的时间片current_minor_frame;

[0029] 步骤S2.8:解锁线程的kernel_stack,切换到该线程的上下文运行;

[0030] 步骤S2.9:结束调度并返回。

[0031] 优选的,所述就绪线程选择步骤中,选择就绪线程时,即vcpu线程或普通任务线程,首先通过当前时间与截止时间的对比判断分区处于新周期还是旧周期;

[0032] 若处于旧周期且启动了虚拟机,即就绪线程为vcpu线程而非普通任务线程,则直接选择当前pcpu就绪队列中首个就绪的vcpu线程进行切换;

[0033] 若处于新周期且启动了虚拟机,即就绪线程为vcpu线程而非普通任务线程,则计算当前新周期的截止时间并切换至当前pcpu就绪队列中的首个就绪的vcpu线程;

[0034] 若处于新周期但没有启动虚拟机,即就绪线程为普通任务线程而非vcpu线程,则

需要重新选择本分区内有效的pcpu并切换至该pcpu的就绪队列中首个就绪的普通任务线程,当切换就绪线程而就绪队列为空时则切换至当前pcpu的空闲idle线程。

[0035] 优选的,所述就绪线程选择步骤具体包括:

[0036] 步骤S3.1:开始;

[0037] 步骤S3.2:比较当前时间与当前pcpu运行线程,即vcpu线程或普通任务线程的截止时间,判断当前线程是否处于新周期中,若是则跳转步骤S3.3,若否则跳转步骤S3.11;

[0038] 步骤S3.3:判断当前是否运行了虚拟机,若运行了虚拟机则当前运行的线程为vcpu线程跳转步骤S3.4,若未运行虚拟机则当前运行线程为普通任务线程跳转步骤S3.5;

[0039] 步骤S3.4:在多个物理处理器核pcpu中寻找有效的pcpu并记录其索引;

[0040] 步骤S3.5:记录此次切换的当前时间作为next_switch_time,和新周期的截止时间作为next_major_frame;

[0041] 步骤S3.6:判断当前pcpu的就绪队列,即vcpu线程或普通任务线程是否为空,不为空则跳转步骤S3.7,空则跳转步骤S3.14;

[0042] 步骤S3.7:判断当前pcpu的就绪队列中首个就绪线程的栈kernel_stack的状态是否为空闲KS_FREE,空闲则跳转步骤S3.8,非空闲则跳转步骤S3.14;

[0043] 步骤S3.8:将当前pcpu就绪队列中首个就绪线程出队;

[0044] 步骤S3.9:判断出队的就绪线程状态是否为结束退出TE_EXITING,是则跳转步骤S3.7,否则跳转步骤S3.10;

[0045] 步骤S3.10:切换到该就绪线程的上下文运行,跳转步骤S3.15结束并返回该线程号;

[0046] 步骤S3.11:比较当前时间与当前pcpu运行线程的截止时间,判断当前线程是否处于旧周期中,若是则跳转步骤S3.12,若否则跳转步骤S3.14;

[0047] 步骤S3.12:判断当前是否运行了虚拟机,若运行了虚拟机则当前运行的线程为vcpu线程跳转步骤S3.6,若未运行虚拟机则当前运行线程为普通任务线程跳转步骤S3.14;

[0048] 步骤S3.13:记录此次切换的当前时间作为next_switch_time,跳转步骤S3.7;

[0049] 步骤S3.14:切换到空闲idle线程;

[0050] 步骤S3.15:结束并返回该线程号。

[0051] 第二方面,提供了一种基于微内核的分时vcpu多核调度系统,所述系统包括:

[0052] 调度初始化模块:将物理处理器核心pcpu进行分组;每个pcpu申请内存空间创建各自的就绪队列、等待队列,并为每个pcpu创建各自的空闲idle线程;

[0053] 分区多核调度模块:在线程调度时,判断当前线程,即vcpu线程或普通任务线程,的时间片是否耗尽,若未耗尽时间片则不能打断;

[0054] 若耗尽时间片,则选取当前pcpu的就绪队列中首个就绪线程,即vcpu线程或普通任务线程进行切换,若当前pcpu的就绪队列为空则直接运行空闲idle线程;

[0055] 就绪线程选择模块:在分区多核调度模块中,选择就绪线程,即vcpu线程或普通任务线程时,通过当前时间与截止时间的对比判断分区处于新周期还是旧周期,根据判断结果选择就绪线程进行切换。

[0056] 优选的,所述调度初始化模块包括:

[0057] 模块M1.1:初始化开始;

- [0058] 模块M1.2:将物理处理器核心pcpu进行分组,一组注册为实时调度策略作为实时分区;另一组注册为非实时调度策略作为普通分区;
- [0059] 模块M1.3:为每个pcpu申请内存空间创建各自的就绪队列和等待队列,并将就绪队列和等待队列地址记录在全局数组cpu_rq[]与cpu_wq[]中;
- [0060] 模块M1.4:为每个pcpu创建各自的空闲idle线程,并将该idle线程标识记录在全局数组cpu_it[]中;
- [0061] 模块M1.5:pcpu开始运行各自的空闲idle线程;
- [0062] 模块M1.6:进行首次调度,当就绪队列不空时,pcpu切换到就绪队列中首个就绪线程,即vcpu线程或普通任务线程运行,否则仍然运行空闲idle线程;
- [0063] 步骤M1.7:结束并返回。
- [0064] 优选的,所述分区多核调度模块具体包括:
- [0065] 模块M2.1:非首次调度开始,若此时pcpu上正在运行空闲idle线程则切换回空闲idle线程后跳转模块M2.9结束并返回idle线程,若此时正在运行vcpu线程或普通任务线程则跳转模块M2.2;
- [0066] 模块M2.2:判断当前运行的线程的状态,thread_state是否为正在运行TS_RUNNING,若为TS_RUNNING则跳转模块M2.3,若不为TS_RUNNING则跳转模块M2.4;
- [0067] 模块M2.3:判断当前运行的线程的时间片是否耗尽,若耗尽则跳转模块M2.4,若未耗尽则无需调度直接跳转模块M2.9结束并返回;
- [0068] 模块M2.4:设置当前线程状态thread_state为TS_INTER,设置当前线程时间片current_minor_frame为0,依据当前线程与物理处理器核pcpu的亲性和affinity加入相应的pcpu的就绪队列中;
- [0069] 模块M2.5:判断当前pcpu的就绪队列是否为空,若为空则当前pcpu切换到空闲idle线程的上下文运行后,跳转模块M2.9结束并返回,不为空则跳转模块M2.6;
- [0070] 模块M2.6:在当前pcpu的就绪队列中选取就绪线程,即vcpu线程或普通任务线程,并将其出队;
- [0071] 模块M2.7:设置KS_LOCKED标志位锁住就绪线程的栈kernel_stack,依据该线程信息设置该线程的时间片current_minor_frame;
- [0072] 模块M2.8:解锁线程的kernel_stack,切换到该线程的上下文运行;
- [0073] 模块M2.9:结束调度并返回。
- [0074] 优选的,所述就绪线程选择模块中,选择就绪线程,即vcpu线程或普通任务线程时,首先通过当前时间与截止时间的对比判断分区处于新周期还是旧周期;
- [0075] 若处于旧周期且启动了虚拟机,即就绪线程为vcpu线程而非普通任务线程,则直接选择当前pcpu就绪队列中首个就绪的vcpu线程进行切换;
- [0076] 若处于新周期且启动了虚拟机,即就绪线程为vcpu线程而非普通任务线程,则计算当前新周期的截止时间并切换至当前pcpu就绪队列中的首个就绪的vcpu线程;
- [0077] 若处于新周期但没有启动虚拟机,即就绪线程为普通任务线程而非vcpu线程,则需要重新选择本分区内有效的pcpu并切换至该pcpu的就绪队列中首个就绪的普通任务线程,当切换就绪线程而就绪队列为空时则切换至当前pcpu的空闲idle线程。
- [0078] 优选的,所述就绪线程选择模块具体包括:

- [0079] 模块M3.1:开始;
- [0080] 模块M3.2:比较当前时间与当前pcpu运行线程的截止时间,即vcpu线程或普通任务线程,判断当前线程是否处于新周期中,若是则跳转模块M3.3,若否则跳转模块M3.11;
- [0081] 模块M3.3:判断当前是否运行了虚拟机,若运行了虚拟机则当前运行的线程为vcpu线程跳转模块M3.4,若未运行虚拟机则当前运行线程为普通任务线程跳转模块M3.5;
- [0082] 模块M3.4:在多个物理处理器核pcpu中寻找有效的pcpu并记录其索引;
- [0083] 模块M3.5:记录此次切换的当前时间作为next_switch_time,和新周期的截止时间作为next_major_frame;
- [0084] 模块M3.6:判断当前pcpu的就绪队列,即vcpu线程或普通任务线程是否为空,不为空则跳转模块M3.7,空则跳转模块M3.14;
- [0085] 模块M3.7:判断当前pcpu的就绪队列中首个就绪线程的栈kernel_stack的状态是否为空闲KS_FREE,空闲则跳转模块M3.8,非空闲则跳转模块M3.14;
- [0086] 模块M3.8:将当前pcpu就绪队列中首个就绪线程出队;
- [0087] 模块M3.9:判断出队的就绪线程状态是否为结束退出TE_EXITING,是则跳转模块M3.7,否则跳转模块M3.10;
- [0088] 模块M3.10:切换到该就绪线程的上下文运行,跳转模块M3.15结束并返回该线程号;
- [0089] 模块M3.11:比较当前时间与当前pcpu运行线程的截止时间,判断当前线程是否处于旧周期中,若是则跳转模块M3.12,若否则跳转模块M3.14;
- [0090] 模块M3.12:判断当前是否运行了虚拟机,若运行了虚拟机则当前运行的线程为vcpu线程跳转模块M3.6,若未运行虚拟机则当前运行线程为普通任务线程跳转模块M3.14;
- [0091] 模块M3.13:记录此次切换的当前时间作为next_switch_time,跳转模块M3.7;
- [0092] 模块M3.14:切换到空闲idle线程;
- [0093] 模块M3.15:结束并返回该线程号。
- [0094] 与现有技术相比,本发明具有如下的有益效果:
- [0095] 1、本发明基于微内核实现,具有更高的灵活性和可靠性;
- [0096] 2、本发明的分时调度算法,有效整合了实时和非实时任务所需的资源;
- [0097] 3、本发明中实时分区调度算法提高了该分区任务调度的实时性。

附图说明

- [0098] 通过阅读参照以下附图对非限制性实施例所作的详细描述,本发明的其它特征、目的和优点将会变得更明显:
- [0099] 图1为本实施例的实时分区VCPUs多核调度算法全流程图;
- [0100] 图2为本实施例的就绪线程选择算法流程图。

具体实施方式

- [0101] 下面结合具体实施例对本发明进行详细说明。以下实施例将有助于本领域的技术人员进一步理解本发明,但不以任何形式限制本发明。应当指出的是,对本领域的普通技术人员来说,在不脱离本发明构思的前提下,还可以做出若干变化和改进。这些都属于本发明

的保护范围。

[0102] 本发明实施例基于灵活的微内核系统架构,在保证安全性、可靠性的前提下,为整合实时与非实时分区的资源,提供了一种基于微内核的分时vcpu多核调度方法,参照图1和图2所示,为运行于微内核之上的虚拟处理器线程分配不同的物理处理器核组,不同物理处理器核分组具有实时性不同的调度策略,即一组为一个分区,不同分区各行其是互不干扰;该分时vcpu多核调度方法具体包括:

[0103] 调度初始化步骤:将物理处理器核心physical cpu(pcpu)进行分组,一组注册为实时调度策略作为实时分区,另一组注册为非实时调度策略(如时间片轮转调度方法或优先级抢占调度方法)作为普通分区。依据用户需求可进行多个分组,每组为一个分区共享同种调度策略,分区间互不干扰,具体地:

[0104] 步骤S1.1:初始化开始;

[0105] 步骤S1.2:将物理处理器核心pcpu进行分组(即分区),一组注册为实时调度策略作为实时分区,并具有本区独有的周期时间片长度,周期内可运行多个线程,但多个线程的时间片长度之和必须小于等于周期长度;另一组注册为非实时调度策略(如时间片轮转调度方法或优先级抢占调度方法)作为普通分区;

[0106] 步骤S1.3:为每个pcpu申请内存空间创建各自的就绪队列和等待队列,并将就绪队列和等待队列地址记录在全局数组cpu_rq[]与cpu_wq[]中;

[0107] 步骤S1.4:为每个pcpu创建各自的空闲idle线程,并将该idle线程标识记录在全局数组cpu_it[]中;

[0108] 步骤S1.5:pcpu开始运行各自的空闲idle线程;

[0109] 步骤S1.6:进行首次调度,当就绪队列不空时,pcpu切换到就绪队列中首个就绪线程(vcpu线程或普通任务线程)运行,否则仍然运行空闲idle线程;

[0110] 步骤S1.7:结束并返回。

[0111] 分区多核调度步骤:在vcpu线程调度时,首先判断当前线程的时间片是否耗尽,未耗尽则不能打断,耗尽时间片时则选取当前pcpu的就绪队列中首个就绪线程进行切换,若当前pcpu的就绪队列为空则直接运行空闲idle线程;具体步骤如下:

[0112] 步骤S2.1:非首次调度开始,若此时pcpu上正在运行空闲idle线程则切换回空闲idle线程后跳转步骤S2.9结束并返回idle线程,若此时正在运行vcpu线程或普通任务线程则跳转步骤S2.2;

[0113] 步骤S2.2:判断当前运行的线程的状态,thread_state是否为正在运行TS_RUNNING,若为TS_RUNNING则跳转步骤S2.3,若不为TS_RUNNING则跳转步骤S2.4;

[0114] 步骤S2.3:判断当前运行的线程的时间片是否耗尽,若耗尽则跳转步骤S2.4,若未耗尽则无需调度直接跳转步骤S2.9结束并返回;

[0115] 步骤S2.4:设置当前线程状态thread_state为TS_INTER,设置当前线程时间片current_minor_frame为0,依据当前线程与物理处理器核pcpu的亲性和affinity加入相应的pcpu的就绪队列中;

[0116] 步骤S2.5:判断当前pcpu的就绪队列是否为空,若为空则当前pcpu切换到空闲idle线程的上下文运行后,跳转步骤S2.9结束并返回,不为空则跳转步骤S2.6;

[0117] 步骤S2.6:在当前pcpu的就绪队列中选取就绪线程(vcpu线程或普通任务线程)并

将其出队；

[0118] 步骤S2.7:设置KS_LOCKED标志位锁住就绪线程的栈kernel_stack,依据该线程信息设置该线程的时间片current_minor_frame;

[0119] 步骤S2.8:解锁线程的kernel_stack,切换到该线程的上下文运行;

[0120] 步骤S2.9:结束调度并返回。

[0121] 就绪线程选择步骤:在分区多核调度步骤中,首先通过当前时间与截止时间的对比判断分区处于新周期还是旧周期,若处于旧周期且启动了虚拟机则直接选择当前pcpu就绪队列中首个就绪线程进行切换,若处于新周期且启动了虚拟机则计算当前新周期的截止时间并切换至当前pcpu就绪队列中的首个就绪线程,若处于新周期但没有启动虚拟机则需要重新选择本分区内有效的pcpu并切换至该pcpu的就绪队列中首个就绪线程,当切换就绪线程而就绪队列为空时则切换至当前pcpu的空闲线程;具体步骤如下:

[0122] 步骤S3.1:开始;

[0123] 步骤S3.2:比较当前时间与当前pcpu运行线程(vcpu线程或普通任务线程)的截止时间,判断当前线程是否处于新周期中,若是则跳转步骤S3.3,若否则跳转步骤S3.11;

[0124] 步骤S3.3:判断当前是否运行了虚拟机,若运行了虚拟机则当前运行的线程为vcpu线程跳转步骤S3.4,若未运行虚拟机则当前运行线程为普通任务线程跳转步骤S3.5;

[0125] 步骤S3.4:在多个物理处理器核pcpu中寻找有效的pcpu并记录其索引;

[0126] 步骤S3.5:记录此次切换的当前时间作为next_switch_time,和新周期的截止时间作为next_major_frame;

[0127] 步骤S3.6:判断当前pcpu的就绪队列(vcpu线程或普通任务线程)是否为空,不为空则跳转步骤S3.7,空则跳转步骤S3.14;

[0128] 步骤S3.7:判断当前pcpu的就绪队列中首个就绪线程的栈kernel_stack的状态是否为空闲KS_FREE,空闲则跳转步骤S3.8,非空闲则跳转步骤S3.14;

[0129] 步骤S3.8:将当前pcpu就绪队列中首个就绪线程出队;

[0130] 步骤S3.9:判断出队的就绪线程状态是否为结束退出TE_EXITING,是则跳转步骤S3.7,否则跳转步骤S3.10;

[0131] 步骤S3.10:切换到该就绪线程的上下文运行,跳转步骤S3.15结束并返回该线程号;

[0132] 步骤S3.11:比较当前时间与当前pcpu运行线程的截止时间,判断当前线程是否处于旧周期中,若是则跳转步骤S3.12,若否则跳转步骤S3.14;

[0133] 步骤S3.12:判断当前是否运行了虚拟机,若运行了虚拟机则当前运行的线程为vcpu线程跳转步骤S3.6,若未运行虚拟机则当前运行线程为普通任务线程跳转步骤S3.14;

[0134] 步骤S3.13:记录此次切换的当前时间作为next_switch_time,跳转步骤S3.7;

[0135] 步骤S3.14:切换到空闲idle线程;

[0136] 步骤S3.15:结束并返回该线程号。

[0137] 本发明实施例提供了一种基于微内核的分时vcpu多核调度方法,基于微内核实现,具有更高的灵活性和可靠性;分时调度算法,有效整合了实时和非实时任务所需的资源;实时分区调度算法提高了该分区任务调度的实时性。

[0138] 本领域技术人员知道,除了以纯计算机可读程序代码方式实现本发明提供的系统

及其各个装置、模块、单元以外,完全可以通过将方法步骤进行逻辑编程来使得本发明提供的系统及其各个装置、模块、单元以逻辑门、开关、专用集成电路、可编程逻辑控制器以及嵌入式微控制器等的形式来实现相同功能。所以,本发明提供的系统及其各项装置、模块、单元可以被认为是一种硬件部件,而对其内包括的用于实现各种功能的装置、模块、单元也可以视为硬件部件内的结构;也可以将用于实现各种功能的装置、模块、单元视为既可以是实现方法的软件模块又可以是硬件部件内的结构。

[0139] 以上对本发明的具体实施例进行了描述。需要理解的是,本发明并不局限于上述特定实施方式,本领域技术人员可以在权利要求的范围内做出各种变化或修改,这并不影响本发明的实质内容。在不冲突的情况下,本申请的实施例和实施例中的特征可以任意相互组合。

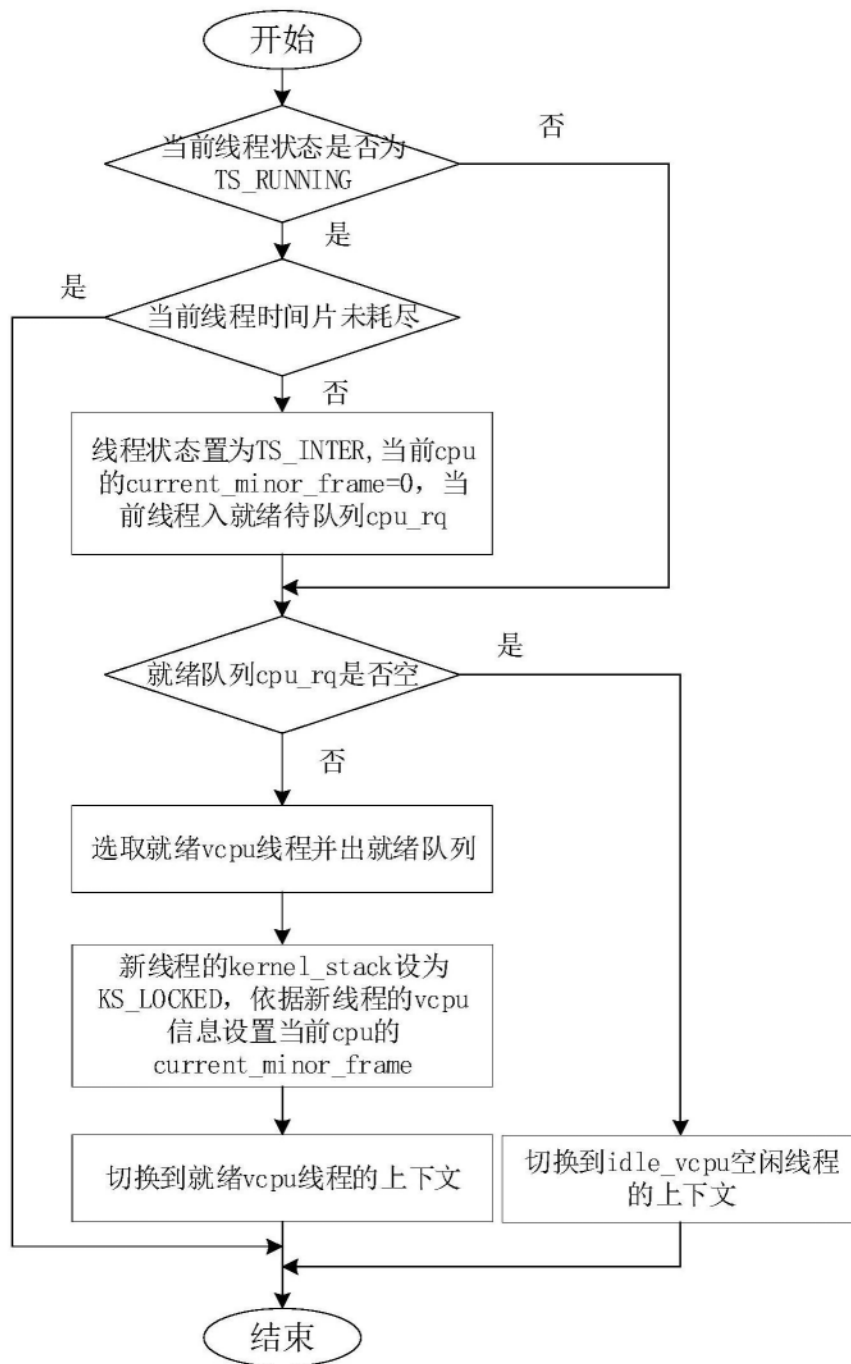


图1

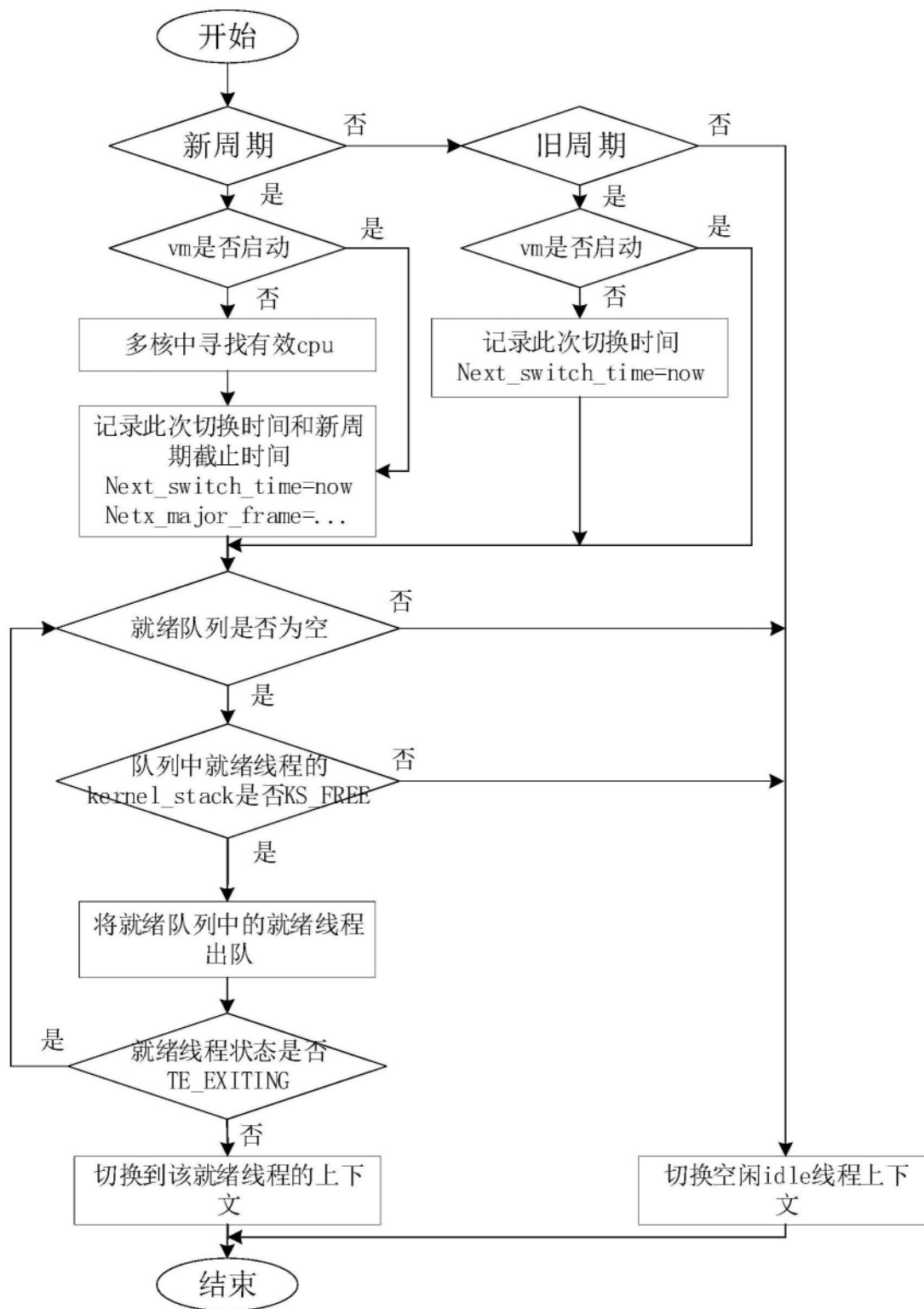


图2