(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 103473135 A
(43)申请公布日 2013.12.25

(21)申请号 201310442373.X

(22)申请日 2013.09.23

(71)申请人 中国科学技术大学苏州研究院
地址 215123 江苏省苏州市工业园区独墅湖高教区仁爱路166号

(72)发明人 吴俊敏 沈恂 赵小雨

(74)专利代理机构 苏州创元专利商标事务所有限公司 32103
代理人 范晴 夏振

(51)Int.Cl.
G06F 9/48(2006.01)
G06F 9/455(2006.01)

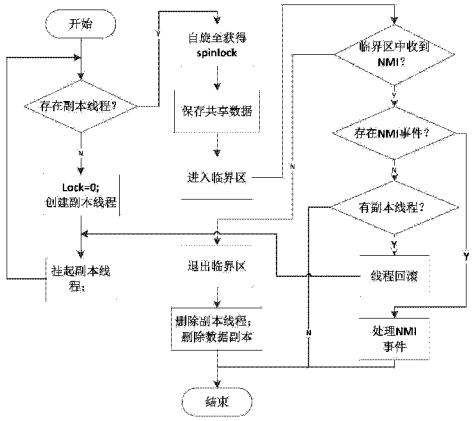
权利要求书1页 说明书6页 附图4页

(54)发明名称

虚拟化环境下自旋锁LHP现象的处理方法

(57)摘要

本发明公开了一种虚拟化环境下自旋锁LHP现象的处理方法，包括以下步骤：(1)客户虚拟机用户态通过系统调用进入客户虚拟机内核，客户虚拟机操作系统内核提供的方法创建当前线程的副本线程，并通过动态申请内核存储空间创建共享数据副本，立即停止副本线程运行；(2)动态检查客户机虚拟操作系统下VCPU对应的客户虚拟机内线程所处的状态，预判是否出现LHP现象；当出现LHP现象时，进行步骤(3)；否则进行步骤(4)；(3)线程回滚操作：删除原线程，运行副本线程，并用共享数据副本值覆盖对应用户地址空间内可能在临界区中被修改的数据进行共享数据还原；(4)删除当前线程的副本线程，并释放共享数据副本占用存储区域，退出临界区后，结束。该方法对宿主机操作系统内调度程序的执行效率鲜有影响，保持了宿主机操作系统内线程调度的高效率。



CN 103473135 A

1. 一种虚拟化环境下自旋锁 LHP 现象的处理方法，其特征在于所述方法中包括以下步骤：

（1）客户虚拟机用户态通过系统调用进入客户虚拟机内核，客户虚拟机操作系统内核提供的方法创建当前线程的副本线程，并通过动态申请内核存储空间创建共享数据副本，立即停止副本线程运行；

（2）动态检查客户机虚拟操作系统下 VCPU 对应的客户虚拟机内线程所处的状态，预判是否出现 LHP 现象；当出现 LHP 现象时，进行步骤（3）；否则进行步骤（4）；

（3）线程回滚操作：删除原线程，运行副本线程，并用共享数据副本值覆盖对应用户地址空间内可能在临界区中被修改的数据进行共享数据还原；

（4）删除当前线程的副本线程，并释放共享数据副本占用存储区域，退出临界区后，结束。

2. 根据权利要求 1 所述的处理方法，其特征在于所述方法步骤（2）中宿主机操作系统判断是否为 VCPU 对应的客户虚拟机内线程是通过预先在宿主机操作系统内核数据结构增设是否为 VCPU 线程的 is_vcpu 字段项来进行的；is_vcpu 字段项初始化为 0；当虚拟机操作系统创建 VCPU 线程时，将 is_vcpu 字段置 1。

3. 根据权利要求 1 所述的处理方法，其特征在于所述方法中宿主机操作系统对 VCPU 线程的干涉，是以发送虚拟 NMI 中断实现；虚拟机操作系统接收到虚拟 NMI 中断后检查系统状态判断出中断来自宿主机操作系统发起的通信，修改后的 NMI 中断处理函数将主导线程进行线程回滚操作并在结束后主动发起 VM_EXIT 切换回宿主机操作系统。

4. 一种虚拟机化环境下自旋锁处理系统，其特征在于所述系统包括用于宿主机操作系统和客户虚拟机操作系统间进行通讯的通讯模块、宿主机操作系统进行线程调度的线程调度模块和客户虚拟机操作系统中的线程回滚模块；所述通讯模块用于宿主机操作系统向客户虚拟机操作系统的 VCPU 发送 NMI 事件以及客户虚拟机操作系统处理 NMI 事件结束后调用 vmcall 指令将 VCPU 切换回宿主机操作系统状态；所述线程调度模块在进行线程调度之前，需要判断相应的线程是否为 VCPU 线程；如果是 VCPU 线程，则通过通讯模块向 VCPU 发送 NMI 事件；当通讯模块切换回宿主机操作系统状态时在宿主机操作系统态运行线程调度程序；所述线程回滚模块用于通讯模块发送完 NMI 事件后，宿主机操作系统将 VCPU 切换到客户虚拟机操作系统状态，调用客户虚拟机操作系统中的 NMI 处理函数进行 NMI 处理；由 NMI 中断处理函数判断 LHP 现象，当出现 LHP 现象时，线程回滚模块进行线程回滚操作。

虚拟化环境下自旋锁 LHP 现象的处理方法

技术领域

[0001] 本发明属于虚拟化技术领域，具体涉及一种虚拟化环境下自旋锁 LHP 现象的处理方法。

背景技术

[0002] 随着计算机技术日新月异的发展，中央处理器经历了从单核高频到单芯多核的不同发展阶段。在单核高频发展阶段，人们着重于提高单一核芯的主频。随着半导体技术的发展遭遇瓶颈，单核处理器的主频止步于约 4GHz。功耗指数级增长、流水线过长等问题使单纯提高核芯的主频已无法带来系统整体性能的提升。多核处理器在处理器芯片上的不同执行内核之间分配任务，提高了单位时钟周期内执行的指令条数。多核多线程处理器技术的发展使得在单核处理器主频提高遭遇瓶颈时，计算机系统仍旧能够满足人们对计算能力日益增长的渴求。

[0003] 系统虚拟化技术的出现，可以让单个处理器核心以分时共享的方式模拟多个核心的并发运行，让单个物理平台上同时运行多个操作系统成为可能，客户程序则在相互隔离的操作系统环境内高效执行、互不干扰，让计算平台基础架构变得简单和高效。运用虚拟化技术还使得应用能够更快部署、性能和可用性得到提升、操作自动进行。以上优点有助于简化 IT 的实施，并降低其拥有和管理成本。

[0004] 自旋锁（spinlock）是用在多个 CPU 系统中的锁机制，当一个 CPU 正访问自旋锁保护的临界区时，临界区将被锁上，其他需要访问此临界区的 CPU 只能忙等待，直到前面的 CPU 已访问完临界区，将临界区开锁。自旋锁上锁后让等待线程进行忙等待而不是睡眠阻塞，而信号量是让等待线程睡眠阻塞。自旋锁的忙等待浪费了处理器的时间，但时间通常很短，在 1 毫秒以下。一个执行单元要想访问被自旋锁保护的共享资源，必须先得到锁，在访问完共享资源后，必须释放锁。如果在获取自旋锁时，没有任何执行单元保持该锁，那么将立即得到锁；如果在获取自旋锁时锁已经有保持者，那么获取锁操作将自旋在那里，直到该自旋锁的保持者释放了锁。

[0005] 系统虚拟化技术已经有了长足的发展，但仍有许多技术难题有待解决。自旋锁（spinlock）的 LHP(lock-holder preemption) 就是众多问题之一。在部署于对称多处理器系统上的虚拟化环境中运行使用自旋锁方式进行同步的多线程程序时会遭遇 LHP 问题。LHP 问题指的是：虚拟化环境中，由于 VCPU(virtual CPU) 可能被虚拟机操作系统管理程序调度出物理处理器，当这个被调度下线的 VCPU 刚好处于客户系统中的临界区状态，且与重新被调度回物理处理器的时间间隔过长时，可能致使运行于其他 VCPU 上的锁竞争者线程自旋忙等时间过长，大量时间片被浪费。本发明因此而来。

发明内容

[0006] 本发明目的在于提供一种虚拟化环境下自旋锁 LHP 现象的处理方法，有效地降低了 VCPU 上的锁竞争者线程自旋忙等时间过长等问题。

[0007] 为了解决现有技术中的这些问题，本发明提供的技术方案是：

[0008] 一种虚拟化环境下自旋锁 LHP 现象的处理方法，其特征在于所述方法中包括以下步骤：

[0009] (1)客户虚拟机用户态通过系统调用进入客户虚拟机内核，客户虚拟机操作系统内核提供的方法创建当前线程的副本线程，并通过动态申请内核存储空间创建共享数据副本，立即停止副本线程运行；

[0010] (2)动态检查客户机虚拟操作系统下 VCPU 对应的客户虚拟机内线程所处的状态，预判是否出现 LHP 现象；当出现 LHP 现象时，进行步骤(3)；否则进行步骤(4)；

[0011] (3)线程回滚操作：删除原线程，运行副本线程，并用共享数据副本值覆盖对应用户地址空间内可能在临界区中被修改的数据进行共享数据还原；

[0012] (4)删除当前线程的副本线程，并释放共享数据副本占用存储区域，退出临界区后，结束。

[0013] 优选的，所述方法步骤(2)中宿主机操作系统判断是否为 VCPU 对应的客户虚拟机内线程是通过预先在宿主机操作系统内核数据结构增设是否为 VCPU 线程的 is_vcpu 字段项来进行的；is_vcpu 字段项初始化为 0；当虚拟机操作系统创建 VCPU 线程时，将 is_vcpu 字段置 1。

[0014] 优选的，所述方法中宿主机操作系统对 VCPU 线程的干涉，是以发送虚拟 NMI 中断实现；虚拟机操作系统接收到虚拟 NMI 中断后检查系统状态判断出中断来自宿主机操作系统发起的通信，修改后的 NMI 中断处理函数将主导线程进行线程回滚操作并在结束后主动发起 VM_EXIT 切换回宿主机操作系统。

[0015] 本发明的另一目的在于提供一种虚拟机化环境下自旋锁处理系统，其特征在于所述系统包括用于宿主机操作系统和客户虚拟机操作系统间进行通讯的通讯模块、宿主机操作系统进行线程调度的线程调度模块和客户虚拟机操作系统中的线程回滚模块；所述通讯模块用于宿主机操作系统向客户虚拟机操作系统的 VCPU 发送 NMI 事件以及客户虚拟机操作系统处理 NMI 事件结束后调用 vmcall 指令将 VCPU 切换回宿主机操作系统状态；所述线程调度模块在进行线程调度之前，需要判断相应的线程是否为 VCPU 线程；如果是 VCPU 线程，则通过通讯模块向 VCPU 发送 NMI 事件；当通讯模块切换回宿主机操作系统状态时在宿主机操作系统态运行线程调度程序；所述线程回滚模块用于通讯模块发送完 NMI 事件后，宿主机操作系统将 VCPU 切换到客户虚拟机操作系统状态，调用客户虚拟机操作系统中的 NMI 处理函数进行 NMI 处理；由 NMI 中断处理函数判断 LHP 现象，当出现 LHP 现象时，线程回滚模块进行线程回滚操作。

[0016] 本发明的目的在于解决虚拟化环境下可能出现的 LHP 问题。通过动态检查 VCPU 对应的客户虚拟机内线程所处的状态，预判是否可能出现 LHP 现象。如判断结果为真，执行线程回滚操作并命令当前线程立刻释放自旋锁使其余竞争线程能够立刻获得自旋锁进入临界区，从而完全消除 LHP 现象、提高客户虚拟机内并行工作负载的运行时间效率。

[0017] 为了针对 LHP 问题实现对自旋锁的优化，本发明技术方案中包括三个模块：客户虚拟机操作系统中的线程回滚模块、宿主机操作系统和客户虚拟机操作系统通讯模块、宿主机操作系统用于 VCPU 对应线程捕捉和调度的线程调度模块。

[0018] 线程回滚模块的主要作用在于线程运行状态的回滚和线程间共享数据的恢复。线

程运行状态回滚的具体方法是在客户虚拟机用户态通过系统调用进入客户虚拟机内核，通过客户虚拟机操作系统内核提供的方法创建当前线程的副本线程，并立即将这个副本线程暂停运行并加以保存。线程间共享数据恢复需要动态申请内核存储空间，保存内核数据结构 mm_struct 中分别起止于 mm->start_data 和 mm->end_data 的已初始化全局变量。这对客户虚拟机内并行负载的编写提出了以下约束，即临界区内只对属于全局已初始化变量的进程共享数据进行修改，且这些全局已初始化变量在临界区以外所有场合只读。需要回滚时用共享数据副本值覆盖对应用户地址空间内可能在临界区中被修改的数据，就完成了数据回滚操作。

[0019] 宿主机操作系统和客户虚拟机操作系统间的通讯模块主要实现宿主机操作系统对 VCPU 线程的干涉。宿主机操作系统对 VCPU 线程的干涉，以发送虚拟中断形式实现。发送的虚拟中断为虚拟 NMI（不可屏蔽中断），NMI 在中断向量中的高优先级保证了 VCPU 在截获中断后立即无条件执行中断服务程序。客户机操作系统接收到中断后检查系统状态判断出中断来自宿主机操作系统发起的通信，修改后的 NMI 中断处理函数将主导线程回滚操作并在结束后主动发起 VM_EXIT 切换回宿主机操作系统。

[0020] 宿主机操作系统 VCPU 对应的线程调度模块，用于宿主机操作系统进行线程调度之前，判断相应的线程是否为 VCPU 线程；如果是 VCPU 线程，则向 VCPU 发送 NMI 事件。发送完 NMI 事件后，宿主机操作系统将 VCPU 切换到客户虚拟机操作系统状态，调用客户虚拟机操作系统中的 NMI 处理函数进行 NMI 处理。

[0021] 线程调度模块的实现基于如下事实：当客户虚拟机操作系统开机后，虚拟机管理程序会创建一个线程作为 VCPU。虚拟机管理程序是 Kvm 软件的一个进程，VCPU 是此进程中的线程。在创建 VCPU 线程前，先在宿主机操作系统中修改 task_struct 结构，增加字段开启对 VCPU 线程识别的支持。具体来说，用 is_vcpu 字段表示线程是否是 VCPU 线程：当此字段值为 1 时，是 VCPU 线程，kvm 字段指向此 VCPU 所属 Kvm 结构；当 is_vcpu 字段值为 0，不是 VCPU 线程，kvm 字段为空。在调用创建 VCPU 的函数时，将新创建的 VCPU 线程对应的 is_vcpu 与 kvm 字段初始化。如此设置后，检查每个的线程的 task_struct 结构中的 is_vcpu 字段就可以得知线程是否是 VCPU 线程。为了提高系统性能，只在宿主机即将真正发生线程调度时才判断此线程是否为 VCPU 线程。在宿主机操作系统实际对线程进行调度之前，判断线程是否为 VCPU 线程，如果是则向 VCPU 发送 NMI 事件。发送完 NMI 事件后，宿主机操作系统将 VCPU 切换到客户虚拟机操作系统状态，以求第一时间调用客户虚拟机操作系统中的 NMI 处理函数。在客户虚拟机态，先检查 VCPU 上运行的用户程序是否处在了临界区中，如果是则说明 LHP 现象发生，需要立刻进行回滚操作。无论有没有进行回滚操作，在结束宿主机发送的 NMI 事件处理过程时，客户虚拟机操作系统都会调用 vmcall 指令再将 VCPU 切换回宿主机操作系统状态。并在宿主机操作系统态运行线程调度代码。

[0022] 相对于现有技术中的方案，本发明的优点是：

[0023] 1. 排除 LHP 现象发生频率小于 10% 的情形，本发明最高可以为并行工作负载争取到约 11% 的运行时间缩减。

[0024] 2. 本发明不在长等现象发生后再对长等进行特征判断，而是通过在客户虚拟机内部对临界区属性进行预设与读取实现完全消除由 LHP 现象引发的长时间自旋忙等。

[0025] 3. 本发明不干涉虚拟机监控器的调度策略，保证了虚拟机调度算法的公平性。

[0026] 4. 本发明对宿主机操作系统内调度程序的执行效率鲜有影响，保持了宿主机操作系统内线程调度的高效率。

[0027] 5. 本发明不依赖对并行负载的静态属性设定，而通过动态采集运行时信息实现对 VCPU 线程状态的判定。

附图说明

[0028] 下面结合附图及实施例对本发明作进一步描述：

[0029] 图 1 是线程回滚操作的总体流程图；

[0030] 图 2 是宿主机操作系统和虚拟机操作系统修改后的 NMI 中断处理函数执行流程图；

[0031] 图 3 是从宿主机视角看 VCPU 线程调度总体流程图；

[0032] 图 4 是从客户虚拟机内线程视角看 VCPU 线程调度总体流程图。

具体实施方式

[0033] 以下结合具体实施例对上述方案做进一步说明。应理解，这些实施例是用于说明本发明而不限于限制本发明的范围。实施例中采用的实施条件可以根据具体厂家的条件做进一步调整，未注明的实施条件通常为常规实验中的条件。

[0034] 实施例

[0035] 本实施例的虚拟机化环境下自旋锁处理系统，包括用于宿主机操作系统和客户虚拟机操作系统间进行通讯的通讯模块、宿主机操作系统进行线程调度的线程调度模块和客户虚拟机操作系统中的线程回滚模块；所述通讯模块用于宿主机操作系统向客户虚拟机操作系统的 VCPU 发送 NMI 事件以及客户虚拟机操作系统处理 NMI 事件结束后调用 vmcall 指令将 VCPU 切换回宿主机操作系统状态；所述线程调度模块在进行线程调度之前，需要判断相应的线程是否为 VCPU 线程；如果是 VCPU 线程，则通过通讯模块向 VCPU 发送 NMI 事件；当通讯模块切换回宿主机操作系统状态时在宿主机操作系统态运行线程调度程序；所述线程回滚模块用于通讯模块发送完 NMI 事件后，宿主机操作系统将 VCPU 切换到客户虚拟机操作系统状态，调用客户虚拟机操作系统中的 NMI 处理函数进行 NMI 处理；由 NMI 中断处理函数判断 LHP 现象，当出现 LHP 现象时，线程回滚模块进行线程回滚操作。

[0036] 如图 1 所示，为虚拟机操作系统内的线程回滚模块的总体流程。

[0037] (1) 线程进入临界区时，创建当前线程的副本，副本线程被创建后立即挂起，原线程继续运行进入临界区；

[0038] (2) 原线程进入临界区前，为临界区内可能修改的数据作好备份，并将数据备份保存；

[0039] (3) 若线程运行在临界区时对应的 VCPU 被调度下线，删除运行中的线程，用挂起的副本线程替代原线程运行，共享数据也随之还原到进入临界区时状态，至此线程回滚完成。

[0040] (4) 回滚结束后 VCPU 下线，spinlock 被释放，某个正占有物理处理器的自旋忙等的竞争线程立刻获得 spinlock，进入临界区；

[0041] (5) 如果持有 spinlock 的线程在临界区内没发生 VCPU 调度，即没有 LHP 现象出

现，删除副本线程和数据备份，原线程正常退出临界区。

[0042] 其中，LHP 现象的判断由宿主机操作系统发起，进入客户虚拟机后，检查 VCPU 上运行的用户程序是否处在了临界区中，如果是则说明 LHP 现象发生，需要立刻进行回滚操作；如果不处于临界区，说明未发生 LHP 现象，无需线程回滚。

[0043] 修改后的 NMI 中断处理函数执行流程如图 2 所示。

[0044] 在客户虚拟机操作系统中对 NMI 中断处理函数进行定制，在接收到 NMI 事件时首先判断所处系统状态：当判断出系统处于无 NMI 可处理且当前线程正处于临界区中时，可以判定出此客户机线程为 VCPU 对应线程，收到的 NMI 事件来自于宿主机操作系统的通信，转入 LHP 现象处理流程；当检测出系统存在 NMI 事件需要处理时，转入正常 NMI 处理流程。LHP 现象处理流程中会先调用线程回滚功能函数，判断线程回滚完毕后，调用 vmcall 这条 vmx 新增指令从客户虚拟机操作系统中返回到宿主机操作系统中去。判断线程是否正处于临界区中无需再在 task_struct 结构中增添字段，只需要检测是否存在副本 mm_struct，即 copy_mmdata 是否为 NULL 即可。此外，vmcall 指令只会在客户虚拟机状态中生效，若在非虚拟化环境下调用这条指令，会给出非法指令信息并退出程序。

[0045] 宿主机操作系统 VCPU 线程调度模块涉及到在宿主机操作系统内识别出 VCPU 线程以及与通信模块配合完整化解一次 LHP 现象。识别 VCPU 线程需要在宿主机操作系统内核数据结构 task_struct 中增设 is_vcpu 字段，并默认初始化为 0。当虚拟机创建 VCPU 线程时，将 is_vcpu 字段置 1。在每次进入宿主机操作系统的线程调度函数时，先对此线程的 is_vcpu 字段进行读取，若判断为 VCPU 线程，按照宿主机操作系统和客户虚拟机操作系统通讯模块所述方式对 VCPU 线程进行干涉。

[0046] 从宿主机视角看 VCPU 线程调度总体流程如图 3 所示。

[0047] 在宿主机操作系统实际对线程进行调度之前，判断线程是否为 VCPU 线程，如果是则向 VCPU 发送 NMI 事件。发送完 NMI 事件后，宿主机操作系统将 VCPU 切换到客户虚拟机操作系统状态，以求第一时间调用客户虚拟机操作系统中的 NMI 处理函数。

[0048] 从客户虚拟机内线程视角看 VCPU 线程调度总体流程如图 4 所示。

[0049] 在客户虚拟机态，先检查 VCPU 上运行的用户程序是否处在了临界区中，如果是则说明 LHP 现象发生，需要立刻进行回滚操作。无论有没有进行回滚操作，在结束宿主机发送的 NMI 事件处理过程时，客户虚拟机操作系统都会调用 vmcall 指令再将 VCPU 切换回宿主机操作系统状态。并在宿主机操作系统态运行线程调度代码。

[0050] 实施例所需硬件环境为：Intel Core i5-2430M 型号处理器加 2GB 物理内存，处理器为单核主频 2.40Ghz 的对称双核处理器。

[0051] 实施例搭建的软件环境为：安装 Ubuntu12.04LTS Linux 作为宿主机操作系统，并用修改后的 3.8.11 版本 Linux 内核替代原发行版内核；安装 qemu-0.14.0 作为虚拟机用户态支持；通过 qemu-kvm 命令行，加载客户虚拟机内核镜像与客户虚拟机根文件系统启动客户虚拟机操作系统，启动命令为：#qemu-system-x86_64 - kernel/boot/vmlinuz-3.8.11 - hda linux-0.2.img - append “root=/dev/sda” - enable-kvm - smp N，其中 - smp N 参数用来设置客户虚拟机对应 VCPU 线程数目为 N。

[0052] 客户虚拟机内核镜像采用 3.8.11 版本 Linux 内核编译制作，客户虚拟机根文件系统使用 qemu 官网提供的精简根文件系统 linux-0.2.img。通过挂载 linux-0.2.img 到本地

目录方式向客户虚拟机操作系统添加动态库与测试程序。

[0053] 客户虚拟机内并行工作负载需要满足约束条件：临界区内只对属于全局已初始化变量的进程共享数据进行修改，且这些全局已初始化变量在临界区以外所有场合只读。

[0054] 上述实例只为说明本发明的技术构思及特点，其目的在于让熟悉此项技术的人是能够了解本发明的内容并据以实施，并不能以此限制本发明的保护范围。凡根据本发明精神实质所做的等效变换或修饰，都应涵盖在本发明的保护范围之内。

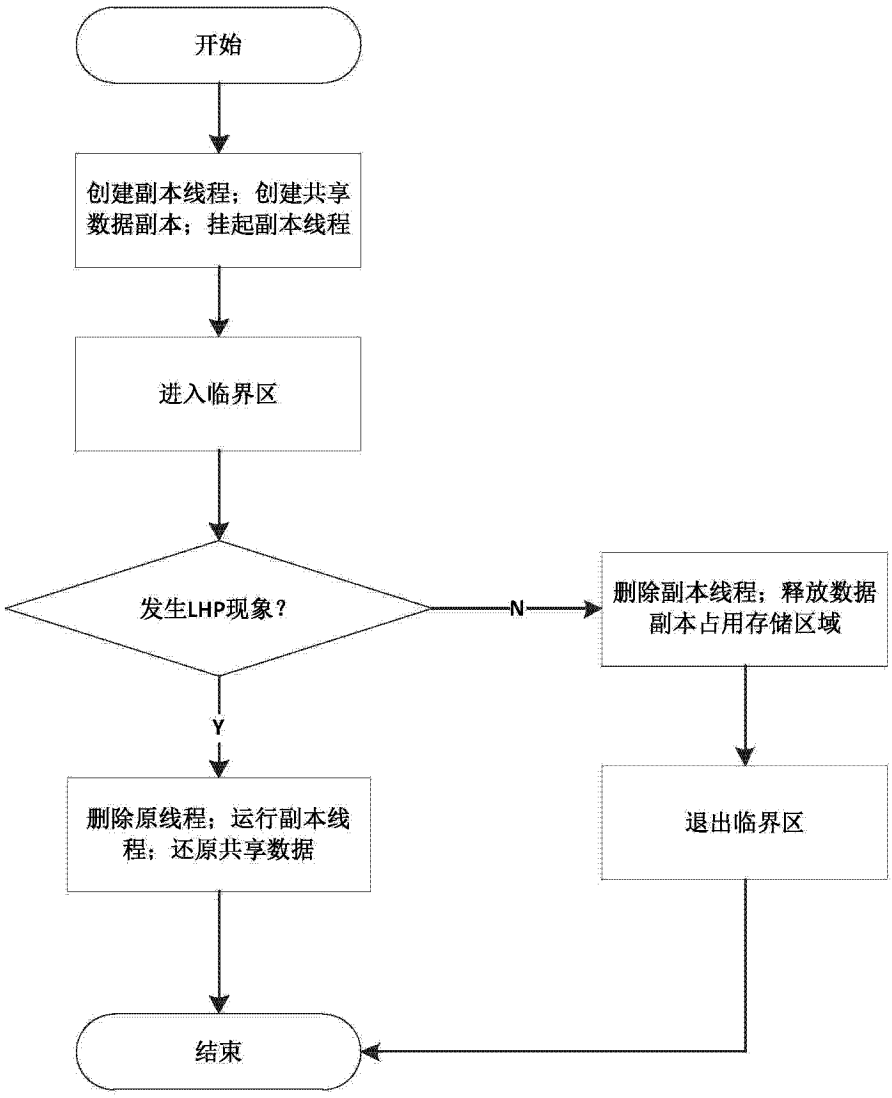

开始
创建副本线程；创建共享数据副本；挂起副本线程
进入临界区
发生LHP现象？
N
Y
删除副本线程；释放数据副本占用存储区域
删除原线程；运行副本线程；还原共享数据
退出临界区
结束

图 1

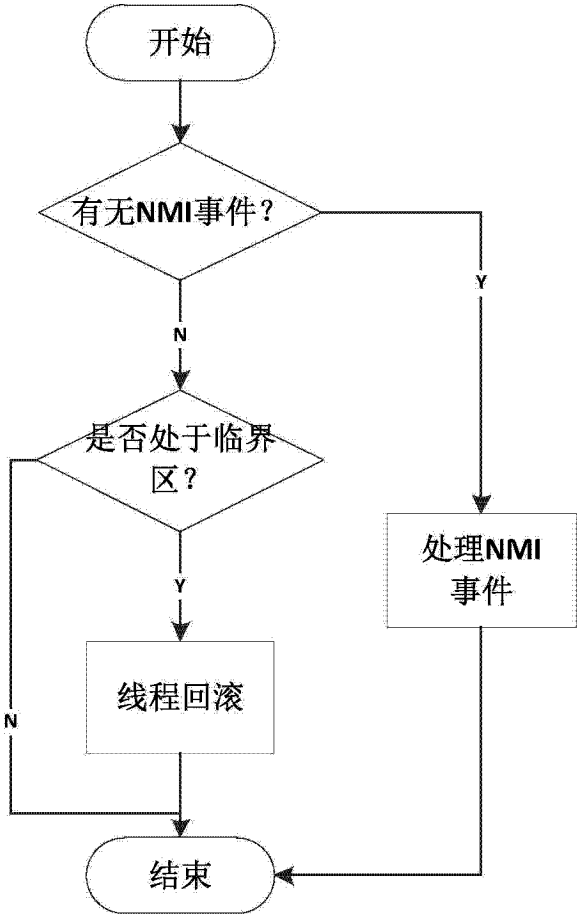


图 2

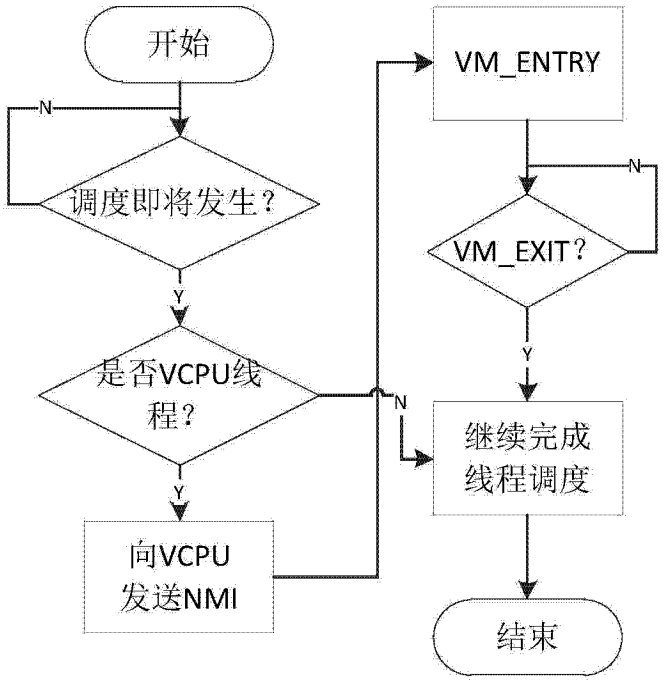

开始
N
调度即将发生？
Y
是否VCPU线程？
Y
向VCPU发送NMI
VM_ENTRY
N
VM_EXIT？
Y
N
继续完成线程调度
结束

图 3

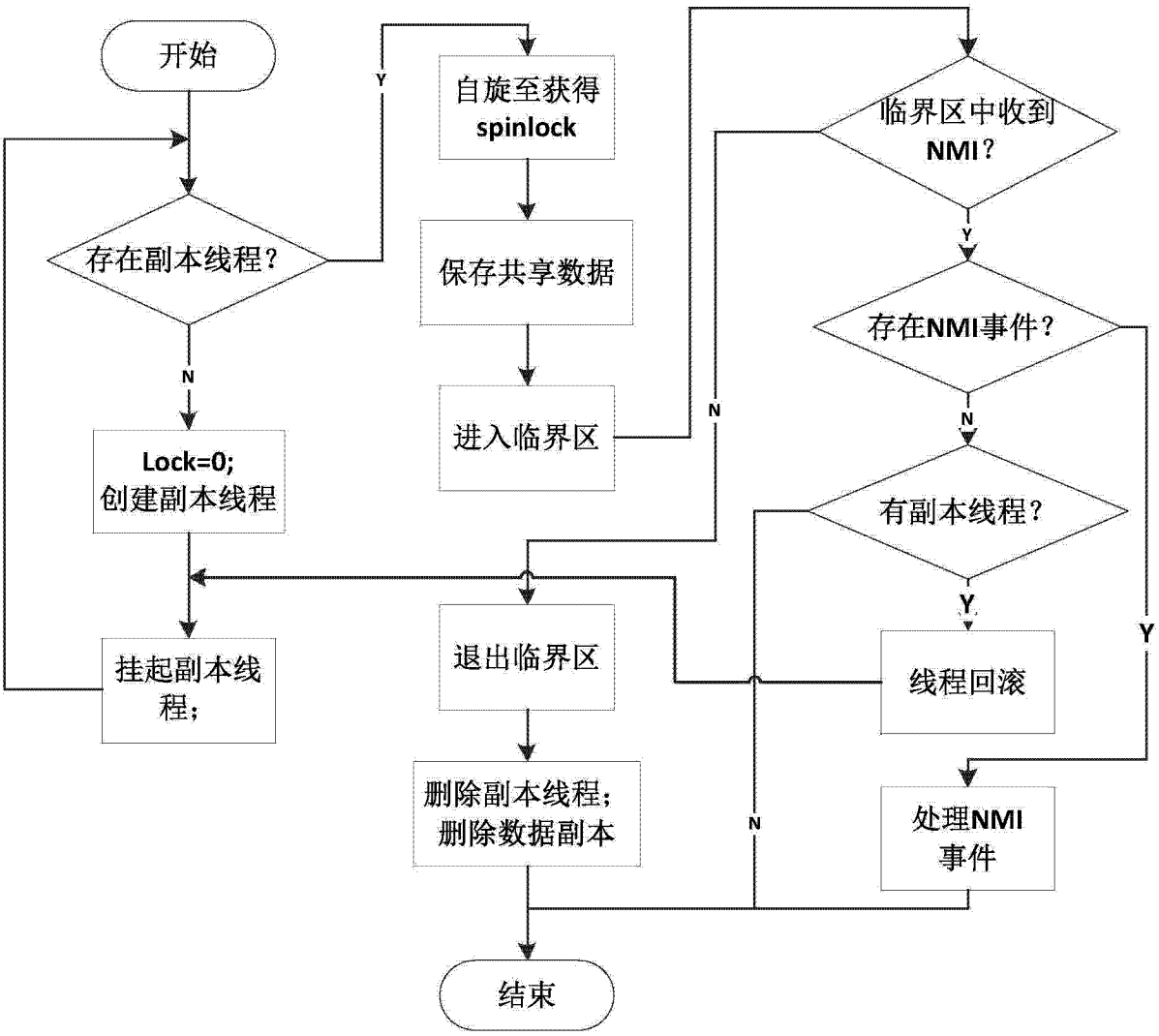

开始
存在副本线程？
Y
N
自旋至获得
spinlock
保存共享数据
进入临界区
Lock=0;
创建副本线程
挂起副本线
程；
退出临界区
删除副本线程；
删除数据副本
结束
临界区中收到
NMI？
Y
N
存在NMI事件？
Y
N
有副本线程？
Y
N
线程回滚
处理NMI
事件
Y

图 4