(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 104679593 B
(45)授权公告日 2017.12.01

(21)申请号 201510113178.1

(22)申请日 2015.03.13

(65)同一申请的已公布的文献号
申请公布号 CN 104679593 A

(43)申请公布日 2015.06.03

(73)专利权人 浪潮集团有限公司
地址 250101 山东省济南市高新区浪潮路1036号

(72)发明人 周恒钊 刘璧怡

(74)专利代理机构 济南信达专利事务所有限公司 37100
代理人 姜明

(51)Int.Cl.
G06F 9/50(2006.01)
G06F 9/48(2006.01)

(56)对比文件
CN 103279391 A,2013.09.04,
CN 103729248 A,2014.04.16,
US 5524077 A,1996.06.04,
CN 101706742 A,2010.05.12,
US 2007/0106879 A1,2007.05.10,

审查员 刘朝兵

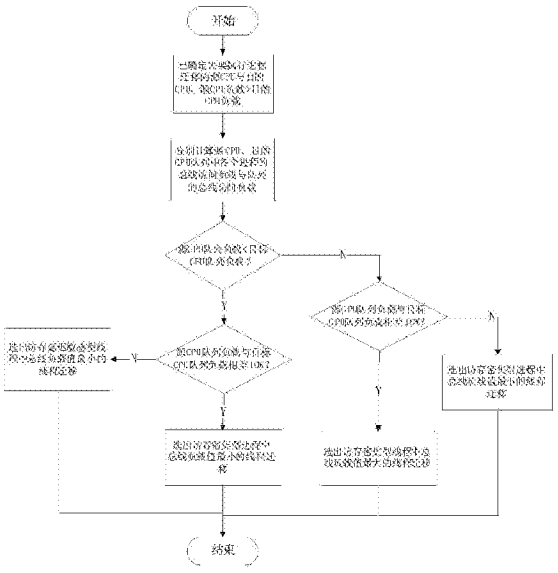
权利要求书1页 说明书5页 附图2页

(54)发明名称

一种基于SMP系统的任务调度优化方法

(57)摘要

本发明公开了一种基于SMP系统的任务调度优化方法，其具体实现过程为：首先进行访存类型划分：按照对于内存或者总线访问的密集程度，将待调度线程划分为访存延迟敏感型与访存密集型两类；对线程带宽访问，即通过处理器内建的硬件性能计数功能获取运行时线程的访问总线速率；进行负载均衡优化，该负载均衡通过调用函数来实现基于调度域的均衡操作；根据上述总线带宽使用情况的SMP任务调度优化策略，该调度优化策略分为两大部分：性能技术采样模块与总线访问负载均衡模块。该一种基于SMP系统的任务调度优化方法与现有技术相比，在不影响原有算法CPU负载均衡效果的基础上优化了总线带宽的使用，提升了总线的有效利用率，实用性强，易于推广。



CN 104679593 B

1.一种基于SMP系统的任务调度优化方法，其特征在于，其具体实现过程为：

首先进行访存类型划分：按照对于内存或者总线访问的密集程度，将待调度线程划分为访存延迟敏感型与访存密集型两类，所述访存延迟敏感型线程是指占用CPU时间较多而访存请求较少的计算型任务，用于区分不同任务的总线访问频度的高低；访存密集型线程则是通过进程平均睡眠时间的计算区分不同任务CPU执行时间的占用比重高低，该访存密集型线程通过内存访问密集度衡量，内存访问密集度为线程访问最末级 Cache 未命中产生访存请求的频度，其使用每千条指令cache丢失数来度量；

对线程带宽访问，即通过处理器内建的硬件性能计数功能获取运行时线程的访问总线速率；

进行负载均衡优化，该负载均衡通过调用load_balance函数来实现基于调度域的均衡操作；

根据上述总线带宽使用情况的SMP任务调度优化策略，该调度优化策略分为两大部分：性能计数采样模块与总线访问负载均衡模块，其中性能计数采样模块以1/T频率执行对线程性能计数信息的采样，并且总是维护最近 Twindow时间内，即采样时间窗口内的采样数据；在每间隔 Twindow时间进行线程内存访问密集度的计算；当内核执行CPU负载均衡操作选出了负载最重的CPU进程就绪队列与轻载的目标队列并且将要执行进程迁移操作时，总线访问负载均衡模块就分别计算源队列与目标队列中进程总线负载，并从源队列挑选进程进行迁移操作。

2.根据权利要求1所述的一种基于SMP系统的任务调度优化方法，其特征在于，所述性能计数采样模块采样过程为：当内核调用fork函数以创建新进程时，initSampleContext函数被调度；该initSampleContext函数用于为新进程初始化性能采样计数的设置，包括写MSR寄存器以设定需要监测的处理器事件类型；sample函数在时钟中断程序的schedule_tick函数中被调用，以T为周期采样执行进程在被创建时就设定好的事件监测计数。

3.根据权利要求2所述的一种基于SMP系统的任务调度优化方法，其特征在于，所述性能计数采样模块中设置有进程控制块，该进程控制块中添加采样函数，所述采样函数记录采样时间窗口内的进程带宽使用情况、进程总线访问权重、采样计数、采样周期内Cache未命中数统计、采样周期内线程执行的指令周期数统计、采样间隔前的计数状态、采样间隔后的计数状态。

4.根据权利要求1所述的一种基于SMP系统的任务调度优化方法，其特征在于，所述总线访问负载均衡模块的具体工作过程为：load_balance函数首先通过find_busiest_group函数在同一级调度域中找到一组最繁忙的CPU组，随后执行find_busiest_queue函数从find_busiest_group找到的最重负载CPU组里选出一个载重负荷最高的CPU；度量一个CPU负荷的标量是等待在该CPU就绪队列中不同优先级进程占用CPU使用比重值的累加和；最后，挑选被迁移的进程。

一种基于SMP系统的任务调度优化方法

技术领域

[0001] 本发明涉及多处理器数据调度技术，具体地说是一种实用性强、基于SMP系统的任务调度优化方法。

背景技术

[0002] 在1985～2000这段时间里，微处理器性能的增长伴随着单处理器主频或者指令级并行度的提高达到了自20世纪50年代后期和60年代初期第一台晶体管计算机诞生以来的最高速度。有赖于集成电路制造工艺的不断精进与提升，处理器上晶体管的集成度得以不断提高，诸多旨在提升指令级并行度的技术被加入到微处理器中，然而这些技术都没有能够改变线程的串行执行模式。通过指令的猜测执行、分支预测与乱序执行等手段可以从串行程序中找到可以用于并行执行的指令，却不能从根本上提升单处理器的并行能力；依靠增加片上集成的晶体管数目可以获得性能提升，却会导致CPU功耗的成倍增大，这一切都表明通过增加复杂度、添加电路和增大功率所能提升的性能正在减少。因此，当多核技术以及多线程技术以较低的复杂度就实现了单处理器上线程级并行后，迅速得以在处理器制造技术中广泛的应用。

[0003] 对称多处理(Symmetrical Multi-Processing，SMP)是指同时拥有多个同构 CPU、CPU间共享同一存储子系统与总线的处理器结构。SMP结构的特点在于多个处理器并行运行操作系统的单一副本并共享对总线、内存和外设资源的访问。SMP感知的操作系统中为每个CPU都设置了进程就绪队列，所有CPU队列上的进程都可以平等地进行访存、响应中断和应答I/O。通过使用多总线代替单总线，或者通过交换机可以使集中共享存储架构支持扩展到更大的对称处理器规模上去。SMP技术在面向高性能服务器和工作站中应用的较为广泛。

[0004] 伴随着处理器和主存储器间速度差距的进一步增大，片外访存操作的长延迟瓶颈，容易导致 SMP 系统中总线整体有效利用率的下降。因而面向上述难题的解决、考虑多核多线程感知的 SMP 线程调度机制的设计一直是操作系统研究中的重要方向。基于上述技术，现提供一种基于SMP系统的任务调度优化方法。

[0005] 发明内容

[0006] 本发明的技术任务是针对以上不足之处，提供一种实用性强、基于SMP系统的任务调度优化方法。

[0007] 一种基于SMP系统的任务调度优化方法，其具体实现过程为：

[0008] 首先进行访存类型划分：按照对于内存或者总线访问的密集程度，将待调度线程划分为访存延迟敏感型与访存密集型两类；

[0009] 对线程带宽访问，即通过处理器内建的硬件性能计数功能获取运行时线程的访问总线速率；

[0010] 进行负载均衡优化，该负载均衡通过调用load_balance函数来实现基于调度域的均衡操作；

[0011] 根据上述总线带宽使用情况的SMP任务调度优化策略，该调度优化策略分为两大

部分:性能计数采样模块与总线访问负载均衡模块,其中性能计数采样模块以1/T频率执行对线程性能计数信息的采样,并且总是维护最近 Twindow时间内,即采样时间窗口内的采样数据;在每间隔 Twindow时间进行线程内存访问密集度的计算;当内核执行CPU负载均衡操作选出了负载最重的CPU进程就绪队列与轻载的目标队列并且将要执行进程迁移操作时,总线访问负载均衡模块就分别计算源队列与目标队列中进程总线负载,并从源队列挑选进程进行迁移操作。

[0012] 所述访存延迟敏感型线程是指占用CPU时间较多而访存请求较少的计算型任务,用于区分不同任务的总线访问频度的高低;访存密集型线程则是通过进程平均睡眠时间的计算区分不同任务CPU执行时间的占用比重高低,该访存密集型线程通过内存访问密集度衡量,内存访问密集度为线程访问最末级 Cache 未命中产生访存请求的频度,其使用每千条指令cache丢失数来度量。

[0013] 所述性能计数采样模块采样过程为:当内核调用fork函数以创建新进程时,initSampleContext函数被调度;该initSampleContext函数用于为新进程初始化性能采样计数的设置,包括写MSR寄存器以设定需要监测的处理器事件类型;sample函数在时钟中断程序的schedule_tick函数中被调用,以T为周期采样执行进程在被创建时就设定好的事件监测计数。

[0014] 所述性能计数采样模块中设置有进程控制块,该进程控制块中添加采样函数,所述采样函数记录采样时间窗口内的进程带宽使用情况、进程总线访问权重、采样计数、采样周期内Cache未命中数统计、采样周期内线程执行的指令周期数统计、采样间隔前的计数状态、采样间隔后的计数状态。

[0015] 所述总线访问负载均衡模块的具体工作过程为:load_balance函数首先通过find_busiest_group函数在同一级调度域中找到一组最繁忙的CPU组,随后执行find_busiest_queue函数从find_busiest_group找到的最重负载CPU组里选出一个载重负荷最高的CPU;度量一个CPU负荷的标量是等待在该CPU就绪队列中不同优先级进程占用CPU使用比重值的累加和;最后,挑选被迁移的进程。

[0016] 本发明的一种基于SMP系统的任务调度优化方法,具有以下优点:

[0017] 该发明的一种基于SMP系统的任务调度优化方法解决了在SM多核多线程的场景下系统总线资源利用率的问题;通过测试验证,优化方案在一定程度上可以提升SMP总线访问带宽的整体利用率;在不影响CPU负载均衡效果的基础上,提高总线的带宽利用率;实用性强,易于推广。

附图说明

[0018] 附图1为本发明的采样流程图。

[0019] 附图2为本发明的进程迁移判断流程图。

具体实施方式

[0020] 下面结合附图和具体实施例对本发明作进一步说明。

[0021] 伴随着处理器和主存储器间速度差距的进一步增大,片外访存操作的长延迟瓶颈,容易导致 SMP 系统中总线整体有效利用率的下降。本发明提出了一种基于SMP系统的

任务调度优化方法，主要解决了在 SMP多核多线程的场景下系统总线资源利用率的问题。

[0022] 本发明的目的是这样实现的，采用基于进程带宽使用的调度思路，如果在进行进程迁移时能够考虑到预先已统计出的进程带宽使用状况，就可以在执行负载均衡的同时优化SMP系统总线带宽的使用。

[0023] 一种基于SMP系统的任务调度优化方法，通过获取线程运行时的性能计数来评估其在最近的采样时间窗口内总线带宽使用状况，同时基于该进程带宽使用的调度思路，如果在进行进程迁移时能够考虑到预先已统计出的进程带宽使用状况，就可以在执行负载均衡的同时优化 SMP 系统总线带宽的使用。通过该线程带宽访问情况的采样机制，利用处理器内建的硬件性能计数功能获取运行时线程的访问总线速率。

[0024] 如附图1、图2所示，其具体实现过程为：

[0025] 一、首先进行访存类型划分：按照对于内存或者总线访问的密集程度，将待调度线程划分为访存延迟敏感型与访存密集型两类。

[0026] 进一步的，该步骤一的详细过程为：按照对于内存或者总线访问的密集程度不同可以将待调度线程划分为访存延迟敏感型与访存密集型两类。访存延迟敏感型线程是指占用CPU时间较多而访存请求较少的计算型任务。尽管总体访存请求的数目较少，访存延迟敏感型线程的性能受到花费额外的指令周期来等待访存结果而非CPU计算操作的延迟影响比较大。执行访存密集型线程时会出现频繁的LLC(Last level Cache)未命中情况，因而大量时间被用于等待未决的访存请求与所请求的数据。这一类型线程的执行性能就很大程度上取决于内存系统与总线的吞吐状况，因为即使其前序内存请求被很快响应，持续不断的后继访存请求又将使层级Cache行失效、打断指令流的连续执行。对于任务进行访存密集型与访存延迟敏感型的划分不同于Linux O(1)调度器中I/O型与处理器消耗型区分的处理。前者旨在区分不同任务的总线访问频度的高低；后者则是通过进程平均睡眠时间的计算区分不同任务CPU执行时间的占用比重高低。

[0027] 这里定义内存访问密集度就是线程访问最末级Cache未命中产生访存请求的频度。可以使用每千条指令cache丢失数(Misses /Thousand instructions,MPKI)来度量这一参数。

[0028] 二、对线程带宽访问，即通过处理器内建的硬件性能计数功能获取运行时线程的访问总线速率。采用了硬件性能计数的在线分析方法(在内核空间边采样计数边分析)来为任务调度提供直接决策依据。

[0029] 三、进行负载均衡优化，无论Linux主动负载均衡或者被动负载均衡都会调用load_balance函数来实现基于调度域的均衡操作。概括load_balance的作用就是将最忙CPU队列的进程迁出以降低处理器间负载的不均衡程度。由于优化设计沿用了 Linux SMP负载均衡的框架，在进行均衡各CPU间工作负载的同时也考虑依照总线访问负载做调度。因此，将考虑总线带宽使用的调度决策判断添加在核心函数load_balance中就可以达成这一目标。

[0030] 四、根据上述总线带宽使用情况的SMP任务调度优化策略，该调度优化策略分为两大部分：性能计数采样模块与总线访问负载均衡模块，其中：

[0031] 性能计数采样模块以1/T频率执行对线程性能计数信息的采样，并且总是维护最近Twindow时间内(采样时间窗口内)的采样数据。在每间隔Twindow时间进行线程内存访问

密集度的计算。

[0032] 采样的时机决定了采样周期。一方面，最简单的考虑就是在时钟中断程序会调用的schedule_tick函数中实现采样，即每1ms执行一次采样。然而值得思考的是每tick都采样带来的开销是否可以接受。另一方面，CFS调度器对线程的调度可以改变线程的运行状态，因此要想合理设置线程运行状态数据的采样周期必须首先了解CFS的调度周期(即调度粒度)的设置。实际调度粒度大小需要用其乘以校正因子$1+\log_2(nr_cpus)$，nr_cpus表示现有的CPU数目。于以上考虑，在时种中断处理函数中而不是CFS调度操作时实现采样可以较精确的保证采样周期性要求。设置采样周期T为3ms而非1ms以避免过频繁的采样造成的开销。

[0033] 采样时间窗口的设置是否合适直接关系到调度决策的正确性。窗口值过小，则无法评估线程连续时间内带宽平均使用状况；窗口值过大，则会因为线程总线访问负载值不能反映较近时刻的带宽使用状况而错过能够预先均衡各CPU总线带宽使用的时机。通过测试与分析，设置采样时间窗口为Twindow=5T=15ms。

[0034] 在进程控制块task_structure中需要添加采样函数。采样函数需要记录采样时间窗口内的进程带宽使用情况、进程总线访问权重、采样计数、采样周期内 L3 Cache未命中数统计、采样周期内线程执行的指令周期数统计、采样间隔前的计数状态、采样间隔后的计数状态。

[0035] 采样流程如附图1所示。每当内核调用fork函数以创建新进程时，initSampleContext函数就将被调度。initSampleContext用于为新进程初始化性能采样计数的设置，主要包括写MSR寄存器以设定需要监测的处理器事件类型。sample函数在时钟中断程序的schedule_tick函数中被调用，以T为周期采样执行进程在被创建时就设定好的事件监测计数。

[0036] 总线访问负载均衡模块的工作过程为：

[0037] 一旦内核执行CPU负载均衡操作选出了负载最重的CPU进程就绪队列与轻载的目标队列并且将要执行进程迁移操作时，总线访问负载均衡模块就分别计算源队列与目标队列中进程总线负载。从源队列挑选进程做迁移时不仅应该满足CPU亲和性与没有正在运行的要求，也应当符合迁移后有利于各CPU总线访问负载趋于均衡的条件。

[0038] Linux原生的load_balance函数首先通过find_busiest_group函数在同一级调度域中找到一组最繁忙的CPU组，随后执行find_busiest_queue函数从find_busiest_group找到的最重负载CPU组里选出一个载重负荷最高的CPU；度量一个CPU负荷的标量是等待在该CPU就绪队列中不同优先级进程占用CPU使用比重值的累加和；最后，挑选被迁移的进程。

[0039] 原有Linux负载均衡算法在挑选被迁移进程时只是考虑两点：

[0040] (1) 满足当前没有运行或刚结束运行。

[0041] (2) 考虑其 CPU 亲和性，可以在与当前队列关联的处理器上执行。

[0042] 本文中进程迁移判断流程如附图2所示，为了均衡总线访问负载，在 load_balance 挑选完最重负荷CPU 后挑选被迁移进程时附加上是否可使迁移后总线访问负载均衡的判断条件与相应处理代码。因此，在不影响 CPU 负载均衡效果的基础上，可以进一步提高总线的带宽利用率。

[0043] 本发明的内容以一个计算内存访问密集度的具体实例来描述实现这一调度优化

策略的执行过程。

[0044] 每当执行线程访问LLC(Last Level Cache)的一行数据未命中时就会向主存控制器请求从主存中加载一个Cache行的数据(Intel架构中层级Cache line size为64Bytes)，主存控制器通过总线将所请求大小的数据更新到未命中的Cache行。

[0045] 如果发生频繁的LLC未命中，记录在该线程下访问总线的带宽就会激增。因此，通过每千条执行指令中的LLC未命中数就可以衡量线程访问带宽的密集度。

[0046] 测试机器存在L3 Cache，因此设计时设定采样L3 Cache 的未命中数。使用 RDMSR 和 WRMSR 指令可以通过读写MSR(Model Specific Register)寄存器可以获取所设置计数的事件信息。

[0047] 上述具体实施方式仅是本发明的具体个案，本发明的专利保护范围包括但不限于上述具体实施方式，任何符合本发明的一种基于SMP系统的任务调度优化方法的权利要求书的且任何所述技术领域的普通技术人员对其所做的适当变化或替换，皆应落入本发明的专利保护范围。

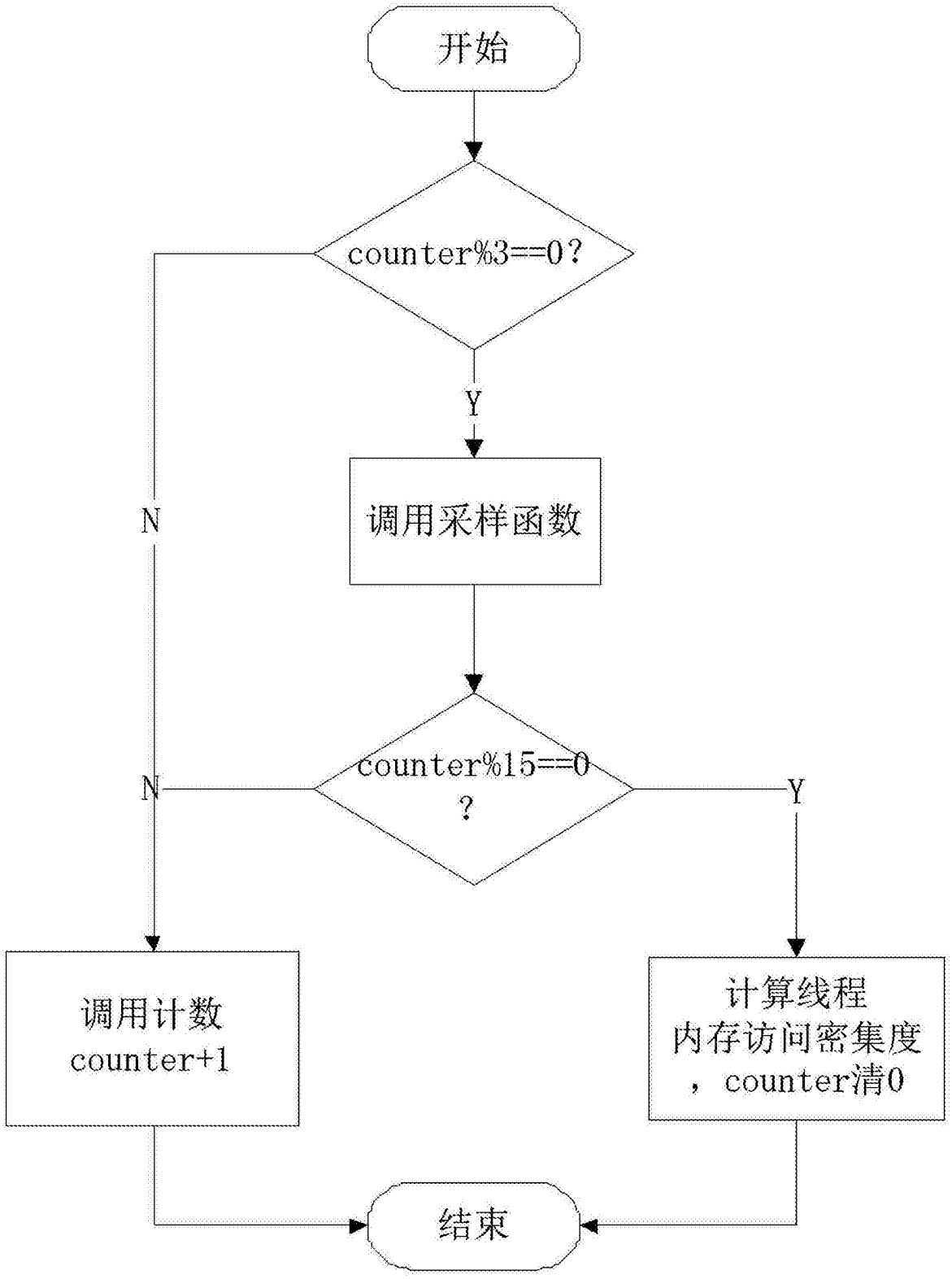

开始
counter%3==0?
Y
N
调用采样函数
counter%15==0
?
N
Y
调用计数
counter+1
计算线程
内存访问密集度
，counter清0
结束

图1

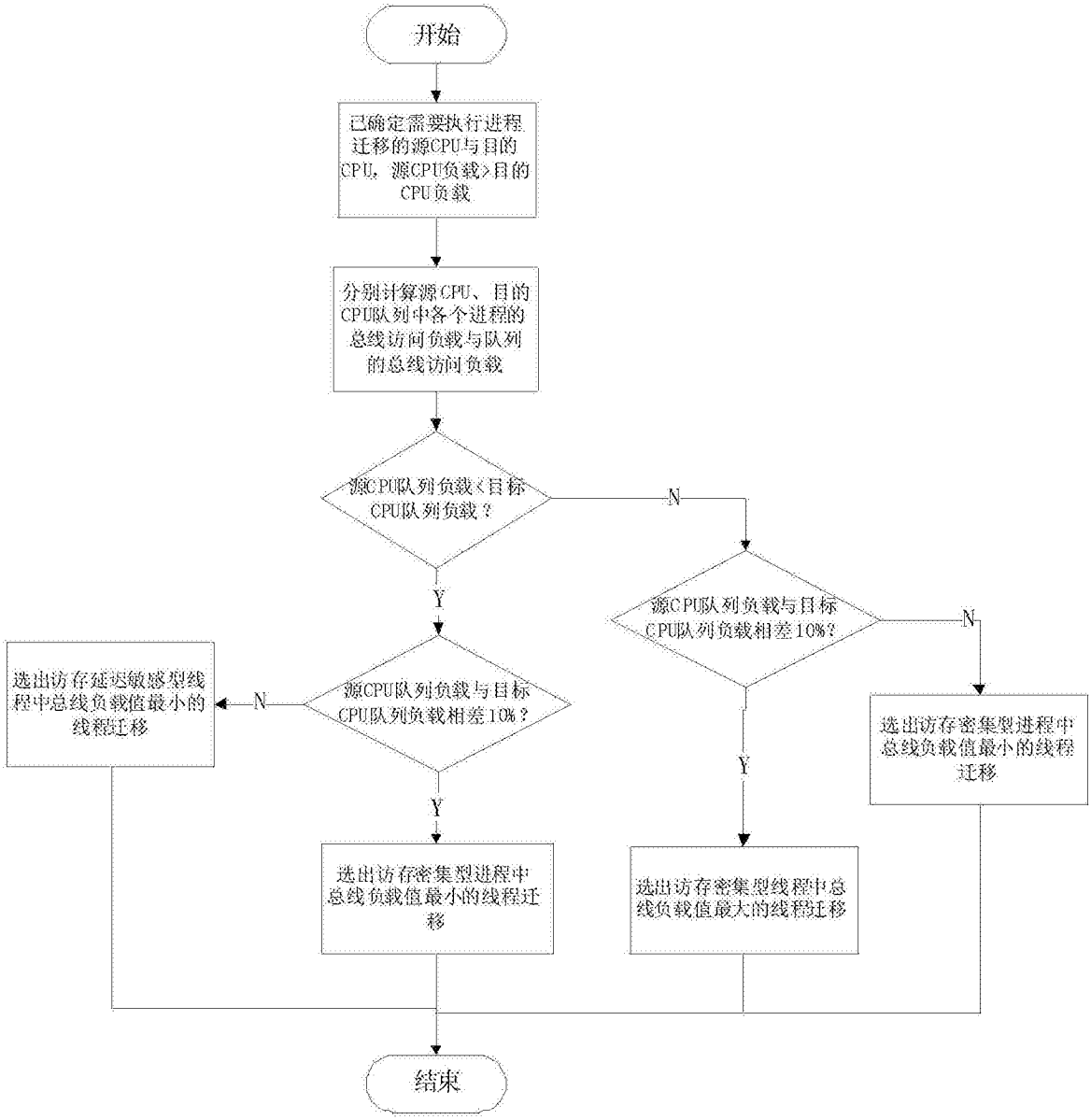

开始
已确定需要执行进程迁移的源CPU与目的CPU，源CPU负载>目的CPU负载
分别计算源CPU、目的CPU队列中各个进程的总线访问负载与队列的总线访问负载
源CPU队列负载<目标CPU队列负载？
N
Y
源CPU队列负载与目标CPU队列负载相差10%？
N
Y
选出访存延迟敏感型线程中总线负载值最小的线程迁移
N
源CPU队列负载与目标CPU队列负载相差10%？
Y
选出访存密集型进程中总线负载值最小的线程迁移
选出访存密集型线程中总线负载值最大的线程迁移
结束

图2