



(12) 发明专利申请

(10) 申请公布号 CN 101923491 A

(43) 申请公布日 2010.12.22

(21) 申请号 201010250104.X

(22) 申请日 2010.08.11

(71) 申请人 上海交通大学

地址 200240 上海市闵行区东川路 800 号

(72) 发明人 过敏意 李阳 王稳寅 丁孟为

杨蓝麒 伍倩 沈耀

(74) 专利代理机构 上海交达专利事务所 31201

代理人 王锡麟 王桂忠

(51) Int. Cl.

G06F 9/50(2006.01)

G06F 9/48(2006.01)

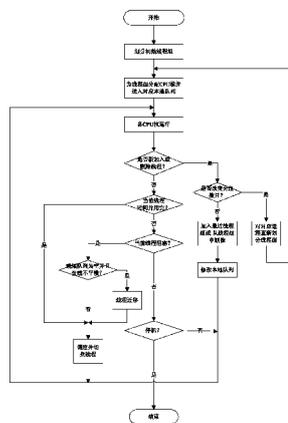
权利要求书 2 页 说明书 4 页 附图 2 页

(54) 发明名称

多核环境下线程组地址空间调度和切换线程的方法

(57) 摘要

一种计算机技术领域中的多核环境下线程组地址空间调度和切换线程的方法,引入了线程分组策略,对潜在的通过 CPU 核分配以及调度次序安排而可以受益的线程进行聚合,减少了调度过程中地址空间切换的次数,提高了 Cache 命中率从而提高了系统吞吐率、提升了系统整体性能;采用的线程分组方法可以灵活地根据硬件平台的特性以及应用特性进行调整,从而能够创造出适应某一特定情况的线程组划分;还可以同其他调度方法结合使用。本发明通过对线程进行分组,并为处理器的每个核配备一个任务队列,实现了具有调度收益的线程依次被调度,具有调度开销小,任务吞吐量大,调度灵活性高的优点。



1. 一种多核环境下线程组地址空间调度和切换线程的方法,其特征在于,包括以下步骤:

第一步,对每个进程所包含的线程进行线程组划分处理,得到若干线程组;

第二步,对线程组进行分配处理,分别得到每个线程组的 CPU 核,并将每个线程组送入对应的本地队列;

第三步,运行各个 CPU 核,当线程被动态创建或删除时,对线程组进行维持处理,得到处理后的线程组;否则,执行第四步;

第四步,当当前线程时间片用完时,调度并切换线程,返回第三步;

否则,当当前线程阻塞,就绪队列为空且负载不平衡时,进行线程迁移,然后进行调度并切换线程,返回第三步;当当前线程阻塞,就绪队列不为空或负载平衡时,直接进行调度并切换线程,返回第三步;当当前线程未阻塞但停机时,线程调度结束;当当前线程未阻塞且未停机时,返回第三步;当阻塞线程恢复为就绪状态时,找到结构队列中向前离该线程最近的就绪线程 u ,并插入线程 u 之前。

2. 根据权利要求 1 所述的多核环境下线程组地址空间调度和切换线程的方法,其特征是,第一步中所述的线程组划分处理,包括以下步骤:

1. 1) 根据 $P_i = \frac{A \times B_i \times C_i}{D}$ 得到每个进程所划分的线程组的组数,其中: P_i 是第 i 个线程所划分的线程组的组数, A 是处理器的核数, B_i 是第 i 个线程的并行因子, C_i 是第 i 个线程的线程数, D 所有线程的总线程数;

1. 2) 将线程指令访问模式差异和线程数据访问模式差异的和作为线程间的差异函数;

1. 3) 采用 K-medoids 聚类方法进行线程组的划分。

3. 根据权利要求 1 所述的多核环境下线程组地址空间调度和切换线程的方法,其特征是,第二步中所述的分配处理,是:在所有的任务队列中,找到不含与线程组 t 同进程的线程组的任务队列集合,并在该任务队列集合中选择长度最短的任务队列所对应的处理器核,作为线程组 t 分配的处理器核。

4. 根据权利要求 1 所述的多核环境下线程组地址空间调度和切换线程的方法,其特征是,第三步中所述的维持处理,包括以下步骤:

2. 1) 当动态创建线程 t 而不改变线程组的组数时,计算线程 t 到其隶属进程的所有线程组中心线程的差异,线程组加入差异最小的线程组,并修改本地队列;

2. 2) 当删除线程 t 而不改变线程组的组数时,直接将线程 t 删除,并修改本地队列;

2. 3) 当动态创建线程 t 或删除线程 t 而改变线程组的组数时,采用第一步的方法,对进程进行重新划分处理,得到若干新的线程组,返回第二步。

5. 根据权利要求 1 所述的多核环境下线程组地址空间调度和切换线程的方法,其特征是,第四步中所述的调度采用先入先出的方式,当调度发生时,选取就绪队列的队头线程作为下一个运行线程,并进行线程切换。

6. 根据权利要求 1 所述的多核环境下线程组地址空间调度和切换线程的方法,其特征是,第四步中所述的负载不平衡,满足以下条件:任务队列中就绪队列为空、持续时间超过阈值 T 、且系统总线程组数超过 CPU 核数。

7. 根据权利要求 1 所述的多核环境下线程组地址空间调度和切换线程的方法,其特征是,第四步所述的线程迁移,是:在就绪队列为空且就绪队列长度大于或者等于 2 的任务队列中选取线程组数量最多的任务队列 L,将 L 中与其中任何线程组都不属于同一进程的线程组 G 迁移到任务队列 L 中。

多核环境下线程组地址空间调度和切换线程的方法

技术领域

[0001] 本发明涉及的是一种计算机技术领域的方法,具体是一种多核环境下线程组地址空间调度和切换线程的方法。

背景技术

[0002] 片上多核处理器(CMP)是一种快速发展的并行解决方案。随着硅技术的发展,在单个晶片上可以集成越来越多的处理器核心。操作系统作为用户与硬件的桥梁,其多核资源的调度策略与调度方法对于系统整体性能的发挥起着至关重要的作用。

[0003] 现代操作系统广泛采用了多线程的计算模型。即进程为计算的静态环境抽象,包括二进制码、地址空间、数据以及资源。线程为进程的动态执行抽象,包括处理机上下文、运行栈以及线程状态。在同一进程下存在一个或多个执行线程,线程为调度的基本单位。

[0004] 线程在进行切换时,会造成显性以及隐性的性能开销。显性开销包括保存上下文时间、执行操作系统调度代码时间以及恢复上下文时间等。隐性开销则包括了由于线程切换导致不同线程访问的高速缓存(Cache)相互影响即Cache污染,从而减少了Cache命中率而导致的性能损失。对于隶属不同进程的线程间切换,还需要进行进程的切换,其中最主要工作就是进程地址空间的切换。地址空间切换也会造成相应的显性和隐性的开销。显性开销包括更换页目录时间,无效化快表(TLB)的时间。对于Intel的x86以及许多复杂指令集架构(CISC)来说,无效化TLB是必须的,因为TLB不记录地址空间信息,地址空间转换意味着原TLB映射的完全失效。而某些精简指令集架构(RISC)如ARM、MIPS提供了具有地址空间标识的TLB,即TLB在进行匹配时不但要匹配虚拟页还要匹配地址空间标识。这样不同地址空间的入口可以在TLB中共存,从而避免了无效化整个TLB的开销。对于地址空间切换的隐性开销来说,不论是否使用地址空间标识的TLB,都会造成很大的性能损失,只是污染比冷启动造成的TLB失效惩罚要少而已。

[0005] 考虑到通用性问题,现在商业操作系统的多核调度方法多为采用基于优先级的调度策略。无论静态优先级还是动态优先级,该策略在进行线程调度时仅考虑到单一线程的属性、执行情况以及当前处理器核的忙碌状态等。传统方法忽略掉或者部分忽略掉了处理器分配与调度次序可能对线程地址空间切换次数、Cache命中率的影响。目前Linux最新2.6版本内核采用了基于优先级的调度策略,并为每一个处理单元分配一个本地就绪队列。然而,调度器在分配任务与负载平衡时并不考虑线程的地址空间信息。

[0006] 经对现有文献检索发现,中国专利申请号为:200810162904,名称为:多核平台下基于硬件计时器与任务队列的调度方法,该技术公开了一种采用在硬件手段,添加多个硬件任务队列与计时器的多核调度方法。该方法旨在借助硬件实现,减少调度开销,但该方法同样忽略了线程的地址空间信息,并且灵活性较差。

发明内容

[0007] 本发明的目的在于克服现有技术的上述不足,提供一种多核环境下线程组地址空

间调度和切换线程的方法。本发明通过对线程进行分组,并为处理器的每个核配备一个任务队列,实现了具有调度收益的线程依次被调度,具有调度开销小,任务吞吐量大,调度灵活性高的优点。

[0008] 本发明是通过以下技术方案实现的,本发明包括以下步骤:

[0009] 第一步,对每个进程所包含的线程进行线程组划分处理,得到若干线程组。

[0010] 所述的线程组划分处理,包括以下步骤:

[0011] 1. 1) 根据 $P_i = \frac{A \times B_i \times C_i}{D}$, 得到每个进程所划分的线程组的组数,其中: P_i 是第 i 个线程所划分的线程组的组数, A 是处理器的核数, B_i 是第 i 个线程的并行因子, C_i 是第 i 个线程的线程数, D 所有线程的总线程数;

[0012] 1. 2) 将线程指令访问模式差异和线程数据访问模式差异的和作为线程间的差异函数;

[0013] 1. 3) 采用 K-medoids 聚类方法进行线程组的划分。

[0014] 第二步,对线程组进行分配处理,分别得到每个线程组的 CPU 核,并将每个线程组送入对应的本地队列。

[0015] 所述的分配处理,是:在所有的任务队列中,找到不含与线程组 t 同进程的线程组的任务队列集合,并在该任务队列集合中选择长度最短的任务队列所对应的处理器核,作为线程组 t 分配的处理器核。

[0016] 第三步,运行各个 CPU 核,当线程被动态创建或删除时,对线程组进行维持处理,得到处理后的线程组;否则,执行第四步。

[0017] 所述的维持处理,包括以下步骤:

[0018] 2. 1) 当动态创建线程 t 而不改变线程组的组数时,计算线程 t 到其隶属进程的所有线程组中心线程的差异,线程组加入差异最小的线程组,并修改本地队列;

[0019] 2. 2) 当删除线程 t 而不改变线程组的组数时,直接将线程 t 删除,并修改本地队列;

[0020] 2. 3) 当动态创建线程 t 或删除线程 t 而改变线程组的组数时,采用第一步的方法,对进程进行重新划分处理,得到若干新的线程组,返回第二步。

[0021] 第四步,当当前线程时间片用完时,调度并切换线程,返回第三步;

[0022] 否则,当当前线程阻塞,就绪队列为空且负载不平衡时,进行线程迁移,然后进行调度并切换线程,返回第三步;当当前线程阻塞,就绪队列不为空或负载平衡时,直接进行调度并切换线程,返回第三步;当当前线程未阻塞但停机时,线程调度结束;当当前线程未阻塞且未停机时,返回第三步;当阻塞线程恢复为就绪状态时,找到就绪队列中向前离该线程最近的就绪线程 u ,并插入线程 u 之前。

[0023] 所述的调度采用先入先出的方式,当调度发生时,选取就绪队列的队头线程作为下一个运行线程,并进行线程切换。

[0024] 所述的负载不平衡,满足以下条件:任务队列中就绪队列为空、持续时间超过阈值 T 、且系统总线程组数超过 CPU 核数。

[0025] 所述的线程迁移,是:在就绪队列为空且就绪队列长度大于或者等于 2 的任务队列中选取线程组数量最多的任务队列 L ,将 L 中与其中任何线程组都不属于同一进程的线

程组 G 迁移到任务队列 L 中。

[0026] 与现有技术相比,本发明的有益效果是:

[0027] 1、减少了显性以及隐性调度开销

[0028] 传统的多核线程调度方法只考虑到单个线程的特征以及各核状态,并不关心线程间的关系以及线程在就绪队列中的排列次序对调度开销的影响。事实上,由于线程切换而导致的潜在的刷新 TLB、TLB 失效以及 Cache 失效都会对系统性能产生很大影响。本方法引入了线程分组策略,对潜在的通过 CPU 核分配以及调度次序安排而可以受益的线程进行聚合,减少了调度过程中地址空间切换的次数,提高了 Cache 命中率从而提高了系统吞吐率、提升了系统整体性能。

[0029] 2、增加了调度灵活性

[0030] 对于传统的调度方法,尤其是硬件实现的调度方法,其调度策略严重依赖于硬件平台以及应用的类型,而本发明的采用的线程分组方法可以灵活地根据硬件平台的特性以及应用特性进行调整,从而能够创造出适应某一特定情况的线程组划分。例如如对于 NUMA 体系结构可以对线程的位置加以考虑从而获得访存局部化更好的线程划分以及 CPU 分配结果。另外,本发明方法还可以同其他方法结合使用,例如在多级队列的每级队列中使用本方法。

附图说明

[0031] 附图 1 是实施例的方法流程图;

[0032] 附图 2 是实施例的任务队列结构图。

具体实施方式

[0033] 以下结合附图对本发明的方法进一步描述:本实施例在以本发明技术方案为前提下进行实施,给出了详细的实施方式和具体的操作过程,但本发明的保护范围不限于下述的实施例。

[0034] 实施例

[0035] 如图 1 所示,本实施例包括以下步骤:

[0036] 第一步,对每个进程所包含的线程进行线程组划分处理,得到若干线程组。

[0037] 所述的线程组划分处理,包括以下步骤:

[0038] 1. 1) 根据 $P_i = \frac{A \times B_i \times C_i}{D}$, 得到每个进程所划分的线程组的组数,其中: P_i 是第 i 个线程所划分的线程组的组数, A 是处理器的核数, B_i 是第 i 个线程的并行因子, C_i 是第 i 个线程的线程数, D 所有线程的总线程数;

[0039] 1. 2) 将线程指令访问模式差异和线程数据访问模式差异的和作为线程间的差异函数;

[0040] 1. 3) 采用 K-medoids 聚类方法进行线程组的划分。

[0041] 第二步,对线程组进行分配处理,分别得到每个线程组的 CPU 核,并将每个线程组送入对应的本地队列。

[0042] 所述的分配处理,是:在所有的任务队列中,找到不含与线程组 t 同进程的线程组

的任务队列集合,并在该任务队列集合中选择长度最短的任务队列所对应的处理器核,作为线程组 t 分配的处理器核。

[0043] 第三步,运行各个 CPU 核,当线程被动态创建或删除时,对线程组进行维持处理,得到处理后的线程组;否则,执行第四步。

[0044] 所述的维持处理,包括以下步骤:

[0045] 2.1) 当动态创建线程 t 而不改变线程组的组数时,计算线程 t 到其隶属进程的所有线程组中心线程的差异,线程组加入差异最小的线程组,并修改本地队列;

[0046] 2.2) 当删除线程 t 而不改变线程组的组数时,直接将线程 t 删除,并修改本地队列;

[0047] 2.3) 当动态创建线程 t 或删除线程 t 而改变线程组的组数时,采用第一步的方法,对进程进行重新划分处理,得到若干新的线程组,返回第二步。

[0048] 第四步,当当前线程时间片用完时,调度并切换线程,返回第三步;

[0049] 否则,当当前线程阻塞,就绪队列为空且负载不平衡时,进行线程迁移,然后进行调度并切换线程,返回第三步;当当前线程阻塞,就绪队列不为空或负载平衡时,直接进行调度并切换线程,返回第三步;当当前线程未阻塞但停机时,线程调度结束;当当前线程未阻塞且未停机时,返回第三步;当阻塞线程恢复为就绪状态时,找到就绪队列中向前离该线程最近的就绪线程 u ,并插入线程 u 之前。

[0050] 所述的调度采用先入先出的方式,当调度发生时,选取就绪队列的队头线程作为下一个运行线程,并进行线程切换。

[0051] 所述的负载不平衡,满足以下条件:任务队列中就绪队列为空、持续时间超过阈值 T 、且系统总线程组数超过 CPU 核数。

[0052] 所述的线程迁移,是:在就绪队列为空且就绪队列长度大于或者等于 2 的任务队列中选取线程组数量最多的任务队列 L ,将 L 中与其中任何线程组都不属于同一进程的线程组 G 迁移到任务队列 L 中。

[0053] 本实施例中每一个 CPU 核对应一个任务队列,具体的任务队列结构如图 2 所示,其中: $TG_i.x$ 表示第 i 号进程的 x 号线程组, $TG_i.x \in TG_i$; $t_i.x$ 表示第 i 号进程的 x 号线程, $t_i.x \in T_i$; T_i 表示第 i 号进程的线程集合。任务队列 L_k 由双向循环链表(结构链表)和单向循环队列(调度队列)组成,图 2 中每一个块代表线程节点,任务队列由共有 3 个线程组($TG_1.3$ 、 $TG_2.5$ 以及 $TG_3.1$)和 9 个线程节点组成。节点组成的环即为结构链表。可见,结构链表中同一线程组内的线程邻接。在没有线程组加入与移出的情况下,结构链表保持不变,即结构链表不受线程就绪与阻塞影响。调度队列也就是线程就绪队列,在附图 2 中用虚线表示,当前执行的线程用实箭头表示,示例中为 $t_3.4$,队头指针指向 $t_3.10$ 。调度队列采用链式实现,在排除了阻塞线程后顺序与结构链表保持一致。节点包含标志位 $block$ 用以区分线程的状态,具体来说, $block$ 为 1 时线程就绪, $block$ 为 0 时线程阻塞。

[0054] 本实施例引入了线程分组策略,对潜在的通过 CPU 核分配以及调度次序安排而可以受益的线程进行聚合,减少了调度过程中地址空间切换的次数,提高了 Cache 命中率从而提高了系统吞吐率、提升了系统整体性能;可以灵活地根据硬件平台的特性以及应用特性进行调整,从而能够创造出适应某一特定情况的线程组划分。

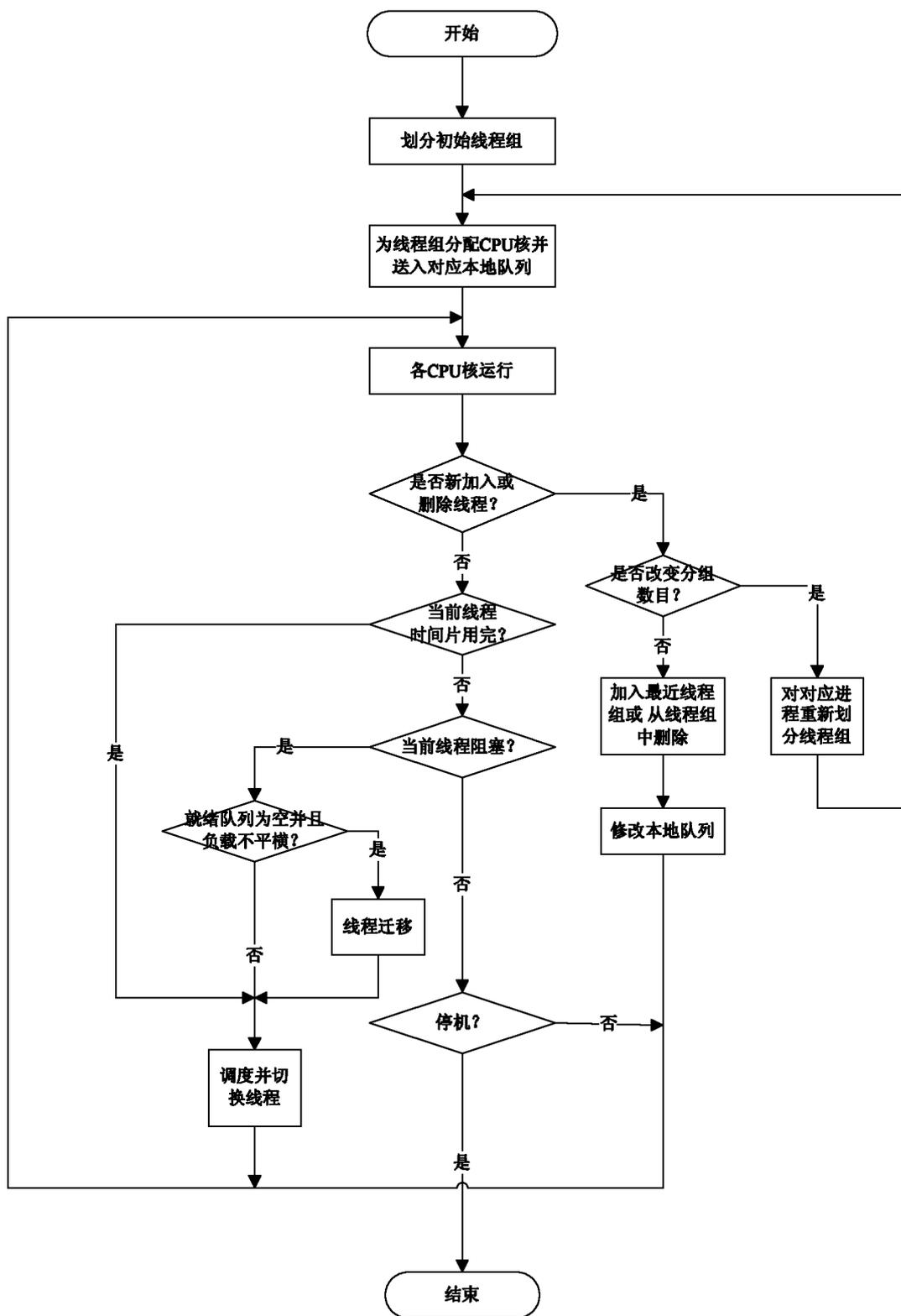


图 1

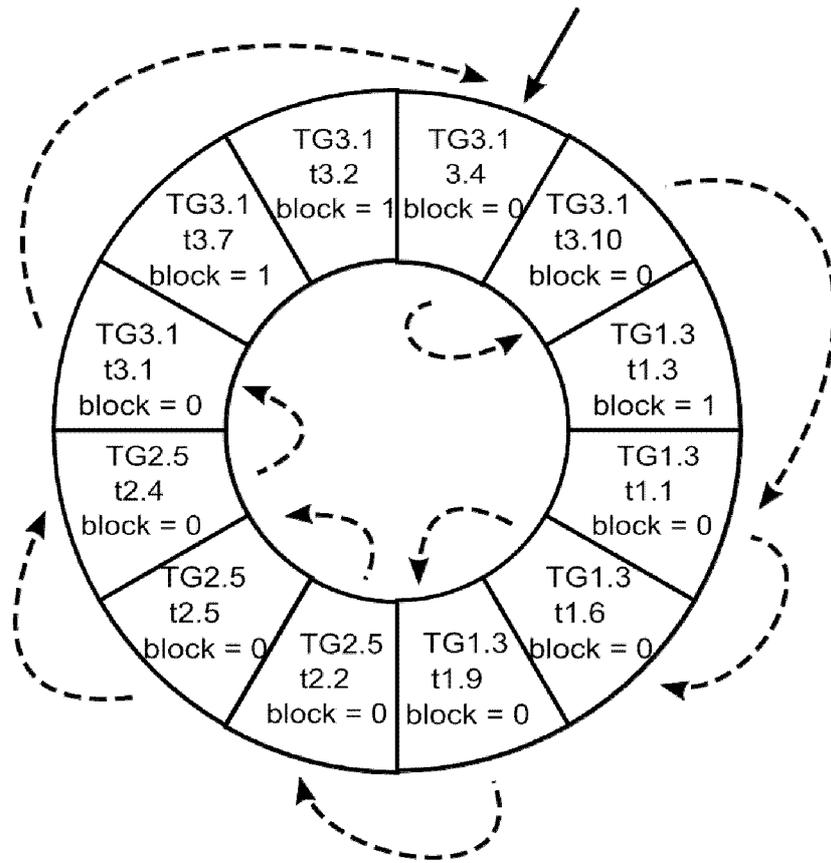


图 2