

(12) 发明专利申请

(10) 申请公布号 CN 102681895 A

(43) 申请公布日 2012. 09. 19

(21) 申请号 201110058641. 9

(22) 申请日 2011. 03. 11

(71) 申请人 北京市国路安信息技术有限公司

地址 100089 北京市海淀区中关村南大街  
32 号 2 号楼 B 座六层 608

(72) 发明人 孙绍钢 李晓勇

(51) Int. Cl.

G06F 9/48(2006. 01)

G06F 9/455(2006. 01)

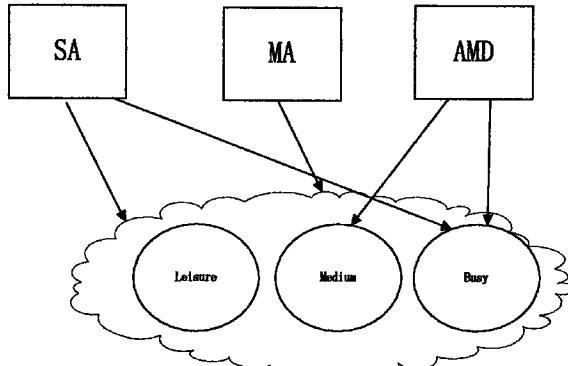
权利要求书 1 页 说明书 4 页 附图 3 页

(54) 发明名称

一种动态自迁移云服务方法

(57) 摘要

本发明涉及一种动态自迁移云服务方法，即云服务可以自动判断虚拟机的状态，当虚拟机满足迁移需求时，就进行动态迁移，在保证服务连续性的同时，提高服务的可用性和可靠性，并节约资源。本发明的效果和优点是：在云服务中，根据当前使用量的多少，虚拟机的开关和工作负荷可以自动的动态调节，当使用量很大时，虚拟机可以被自动的迁移到多台物理机上去，保证服务的可用性；而当使用量变小时，可以将多台物理机上的多个虚拟机上的服务动态迁移到一台物理机的虚拟机上，将其他的物理机关闭，从而达到节省资源的效果。



1. 一种动态自迁移云服务的方法,其特征在于,

a、动态自迁移云服务包括检测模块 SA, 迁移权威 MA, 迁移后检测权威 AMD, 闲队列 {leisure} , 中间队列 {medium} 以及忙队列 {busy} ;

b、迁移的情形分为两种情况,第一种是负载大的机器向负载小的机器迁移,另一种情况是多个负载小的机器向一台机器迁移;

c、迁移算法也分为两种,分散迁移算法和聚合迁移算法;

d、检测模块 SA 对所有物理机进行监测,当发现以下情况时,激活 MA 启动迁移程序:

分散算法:

移出源选择:当发现一台机器的负载超过  $m$ ,则将其放入 {busy} 队列,若其在该队列的时间超过  $t$  分钟,则考虑进行迁移,查看该机器上虚拟机的数量  $V_m$ ,如果  $V_m > 1$ ,则挑选虚拟机进行迁移,计算迁移权值  $MQ$ ,  $MQ = M_{cpu}/M_{mem}*I/O*t$ ,选择  $MQ$  最小的虚拟机进行迁移;

移出目的选择:首先从 {medium} 队列中进行选择,匹配移出源的 CPU、内存、I/O 等数据,如果找到合适的机器,则进行迁移过程,将移出源迁移到这台机器上,否则,从 {leisure} 队列中进行查找,找到合适的目的机器;

迁移时:迁移时采用 Iterative precopy 算法,尽可能的减少迁移造成的服务中断时间,迁移时首先迁移方向 MA 申请迁移,MA 同意后,执行 Iterative precopy 算法,将 VM 迁移进来;

迁移后再判断:迁移完成后持续对移出目的进行监控,如果其没有进入 {busy} 队列,则迁移成功,否则说明迁移到这台机器是不合理的,首先进行回滚操作,恢复到迁移前的状态,重新选择迁移目的,回到步骤 2 进行再迁移;

聚合算法:

检测模块 SA 从 {leisure} 队列中进行监测,发现有物理机在超过  $t_1$  的时间内的,都处于 {leisure} 队列,则将这些物理机挑出,准备进行聚合迁移,从这些物理机中挑选迁移权值  $MQ$  最小的物理机  $m_1$ ,将这些物理机上的 VM 都迁移到  $m_1$  上,迁移完成后将其他不用的物理机全部休眠掉,以节省资源;

迁移完成后进行再判断,如果  $m_1$  没有进入 {busy} 队列,则迁移成功,否则说明迁移到这台机器是不合理的,则进行回滚操作,回到初始状态再进行再迁移,重新选择目的,进行按照 1、2 步骤迁移。

## 一种动态自迁移云服务方法

### 技术领域

[0001] 本发明涉及云计算安全技术领域,特别是云计算中根据虚拟机的状态自动进行虚拟机的动态迁移的方法。

### 背景技术

[0002] 虚拟技术是实现云计算的一个重要的技术。为了保证云服务的高可用性和可靠性,虚拟机的迁移成为研究的一个热点问题。虚拟机迁移目前存在三种方法:Stop-and-copy(S-C)、Demand-migration(D-M) 和 Iterative precopy(I-P)。其中 Stop-and-copy(S-C) 在迁移之前首先将要迁移的虚拟机停止,然后进行传输,迁移完成后再进行恢复。需求迁移(Demand-migration(D-M)),需求迁移时,首先进入很短的 stop-and-copy 阶段,将原虚拟机停止,将重要的核心数据结构首先传递到目的机,然后就将目的虚拟机打开,开始对外提供服务,其他的页在通过网络进行传输。这种迁移方式的与 stop-and-copy 相比的好处是系统停止服务的时间很短,但是迁移的总时间会变长,并且在实践中迁移后的性能会变得不可接受,直到相当多的页都被传过来之后。ChristopherClark, Keir Fraser, Steven Hand 提出 Iterative precopy(I-P) 迁移算法,采用预拷贝(pre-copy)迁移方法,在停止服务之前,首先进行 kernel 的预拷贝,将服务必须的部分先拷贝过来,然后将服务开启,其他需要迁移的部分再慢慢的传输过来。这种方式大大的减少了服务停止的时间,提高了迁移时服务的连续性。

[0003] 面向虚拟计算环境的负载均衡手段则是迁移虚拟机,迁移粒度大,迁移时传输的数据量也大,因而迁移开销是不可忽略的。Sandpiper 实现了热点检测算法用来决定何时迁移虚拟机,一个热点迁移算法用来决定怎么迁移。VMware 在他们的 VirtualCenter 管理软件中加入了对操作系统的支持。VMwareDistributed Resource Scheduler(DRS) 是一种运用虚拟环境中可获得的资源来分配和平衡计算容量的工具。VMware DRS 跨资源池不间断地监控资源利用率,并根据反映业务需要和不断变化的优先级的预定义规则,在多台虚拟机之间智能地分配可用资源。如果有一个或多个虚拟机的工作量大幅度变化,VMware DRS 会在物理服务器之间重新分配虚拟机,通过 VMware VMotion 将虚拟机实时迁移到不同的物理服务器,是以对最终用户完全透明的方式完成的。如果总体工作量减少,一些物理服务器可以暂时关闭。

[0004] 目前的研究多集中在迁移过程中的动态迁移方法,像 VMware、Xen 都已经实现了虚拟机的动态迁移,但是这些迁移过程都需要人工的参与,由管理人员来进行判断和操作,根据虚拟机的状态来选择是否需要迁移,并选择迁移目的地,而对于自动进行迁移前虚拟机状态的判断,是否需要迁移,以及迁移后的再判断,判断该次迁移是否达到了迁移时的目的,则没有相关的研究。

### 发明内容

[0005] 本发明的目的就是针对存在的上述问题,提出一种可以动态自迁移云服务的方

法,即云服务可以自动判断虚拟机的状态,当虚拟机满足迁移需求时,就进行动态迁移,在保证服务连续性的同时,提高服务的可用性和可靠性,并节约资源。

[0006] 本发明的目的是通过如下技术方案来实现的。

[0007] 动态自迁移云服务包括检测模块 SA,迁移权威 MA,迁移后检测权威 AMD,闲队列 {leisure},中间队列 {medium} 以及忙队列 {busy}。迁移的情形分为两种情况,第一种是负载大的机器向负载小的机器迁移,另一种情况是多个负载小的机器向一台机器迁移。相应的,迁移算法也分为两种,分散迁移算法和聚合迁移算法。

[0008] 检测模块 SA 对所有物理机进行监测,当发现以下情况时,激活 MA 启动迁移程序:

[0009] 分散算法:

[0010] 1、移出源选择:当发现一台机器的负载超过  $m$ ,则将其放入 {busy} 队列,若其在该队列的时间超过  $t$  分钟,则考虑进行迁移,查看该机器上虚拟机的数量  $V_m$ ,如果  $V_m > 1$ ,则挑选虚拟机进行迁移,计算迁移权值  $MQ$ , $MQ = Mcpu/Mmem*I/O*t$ 。,选择  $MQ$  最小的虚拟机进行迁移。

[0011] 2、移出目的选择:首先从 {medium} 队列中进行选择,匹配移出源的 CPU、内存、I/O 等数据,如果找到合适的机器,则进行迁移过程,将移出源迁移到这台机器上。否则,从 {leisure} 队列中进行查找,找到合适的目的机器。

[0012] 3、迁移时:迁移时采用 Iterative precopy 算法,尽可能的减少迁移造成的服务中断时间。迁移时首先迁移方向 MA 申请迁移,MA 同意后执行 Iterativeprecopy 算法,将 VM 迁移进来。

[0013] 4、迁移后再判断:迁移完成后持续对移出目的进行监控,如果其没有进入 {busy} 队列,则迁移成功,否则说明迁移到这台机器是不合理的,首先进行回滚操作,恢复到迁移前的状态,重新选择迁移目的,回到步骤 2 进行再迁移。

[0014] 聚合算法:

[0015] 1、检测模块 SA 从 {leisure} 队列中进行监测,发现有物理机在超过  $t_1$  的时间内的,都处于 {leisure} 队列,则将这些物理机挑出,准备进行聚合迁移。从这些物理机中挑选迁移权值  $MQ$  最小的物理机  $m_1$ ,将这些物理机上的 VM 都迁移到  $m_1$  上,迁移完成后将其他不用的物理机全部休眠掉,以节省资源。

[0016] 2、迁移完成后进行再判断,如果  $m_1$  没有进入 {busy} 队列,则迁移成功,否则说明迁移到这台机器是不合理的,则进行回滚操作,回到初始状态再进行再迁移,重新选择目的,进行按照 1、2 步骤迁移。

[0017] 本发明的效果和优点是:在云服务中,根据当前负载量的多少,物理机和虚拟机的开关和工作负荷可以自动的动态调节,当负载量很大时,虚拟机可以被自动的动态迁移到多台物理机上去,保证服务的可用性;而当负载量变小时,可以将多台物理机上的多个虚拟机上的服务动态迁移到一台物理机的虚拟机上,将其他的物理机关闭,从而达到节省资源的效果。

## 附图说明

[0018] 图 1 是本发明的整体结构图;

[0019] 图 2 是本发明的分散方法流程图;

[0020] 图 3 是本发明的聚合方法流程图。

## 具体实施方式

[0021] 下面结合附图和具体实施方式对本发明作进一步的说明。

[0022] 实施例之一：

[0023] 如图 1 所示,动态自迁移云服务的整体结构,包括检测模块 SA,迁移权威 MA,迁移后检测权威 AMD,闲队列 {leisure},中间队列 {medium} 以及忙队列 {busy}。SA 负责对各个物理机进行监控,记录各物理机的状态,按照机器 CPU、I/O、内存等使用状态对机器进行分类,CPU、I/O、内存中的使用率有一个超过阀值 Fmax,则划入 {busy} 组;CPU、I/O、内存的使用率都小于 Fmin,则划入 {leisure} 队列,其他的划入 {medium} 队列。当满足迁移算法时,通知迁移权威 MA。MA 负责对满足条件的虚拟机进行动态迁移,使用 Iterative precopy 算法进行迁移,保证业务的连续性。迁移完成后由 AMD 进行迁移后物理机状态的监控,如果发现迁移没有达到迁移前的目的,如出现负载过大、过小的情况,影响到了服务的性能,则进行迁移回滚,恢复到迁移前的状态,重新进行判断。

[0024] 实施例之二：

[0025] 如图 2 所示,当物理机负载过大时,分散迁移算法实施例 :SA 对 {busy} 队列进行监控,当发现某个物理机 p1 处于 busy 队列中的时间超过了 t,则通知 MA,需要对 p1 进行分散迁移。MA 接到通知后,首先选择迁移源,对 p1 上的虚拟机进行分析,选择迁移权值 MQ 最大的进行迁移, $MQ = Mcpu/Mmem*MI/0*t$ ,其中 Mcpu 为 CPU 使用率,Mmem 为内存使用率,MI/0 为 I/O 使用率,t 为该虚拟机从上次迁移完成到现在的时间间隔。选择好迁移源之后,接着选择迁移目的机 m1,从 {leisure} 队列中进行选择,按照进入 {leisure} 队列的时间顺序,选择最早进入队列的虚拟机 v1 为迁移目的 m1,若 {leisure} 队列中没有了机器,则从关闭的机器中选择一台机器激活,将其作为迁移目的 m1。选择好迁移源和迁移目的后,MA 使用 Pre-copy 方法来进行虚拟机的动态迁移,迁移时首先迁移方向 MA 申请迁移,MA 同意后执行 Iterative precopy 算法,将虚拟机迁移进来,迁移完成后通知 AMD。AMD 首先对迁移目的机 m1 进行检测,如果迁移完成后 m1 进入了 {busy} 队列,则进行迁移回滚,取消所有的迁移操作,将 v1 迁移回 p1,重新选择迁移目的机 m2 进行迁移。如果 m1 没有进入 {busy} 对列,则对 p1 进行检测。若 p1 仍处于 {busy} 队列,则通知 MA 继续迁移 p1 上的虚拟机,反之则完成此次的迁移任务。

[0026] 实施例之三：

[0027] 如图 3 所示,当多个物理机的负载都很低时,进行聚合迁移实施例 :SA 对 {leisure} 队列进行检测,当发现其中的物理机 p1、p2、p3...pn 在 {leisure} 中的时间超过 t 时,则通知 MA,将这些物理机进行聚合迁移操作。MA 在这些物理机中选择迁移权值 MQ 最小的物理机 m1 作为迁移目的地,将其他物理机上的虚拟机一个一个迁移到 m1 上,迁移时的方法同实施例之二中的迁移时相同,一个虚拟机迁移完成后,由 AMD 对 m1 进行检测,若 m1 进入 {busy} 队列,则进行迁移回滚操作,恢复迁移前的状态,重新选择迁移目的进行迁移,否则,迁移成功,继续下一个虚拟机的迁移。当一台物理机上的虚拟机都被迁移走之后,则将这台虚拟机关闭。当 pn 处理完成后,这次的迁移过程全部完成。

[0028] 本技术领域中的相关技术人员应当熟悉到,以上所述实施例仅是用来说明本发明

的目的,而并非用作对本发明的限定,只要在本发明的实质范围内,对上述实施例所做的变化、变型都将落在本发明的权利要求范围内。

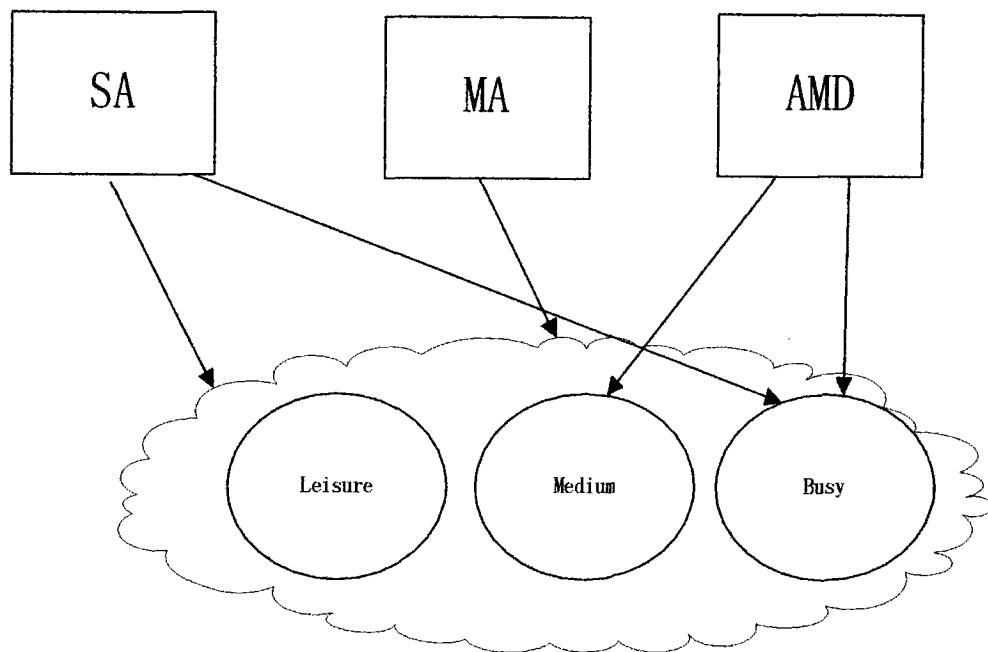


图 1

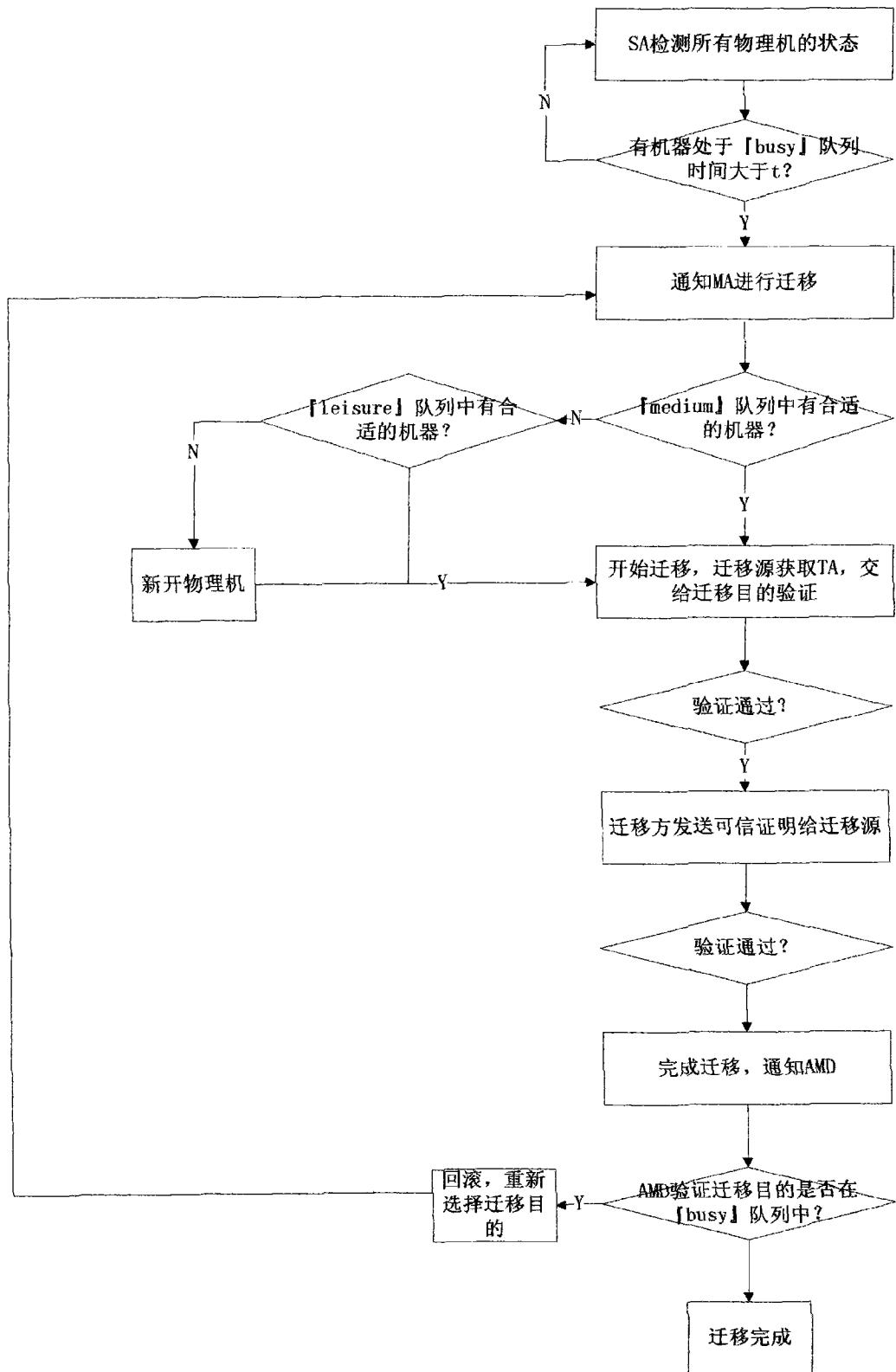


图 2

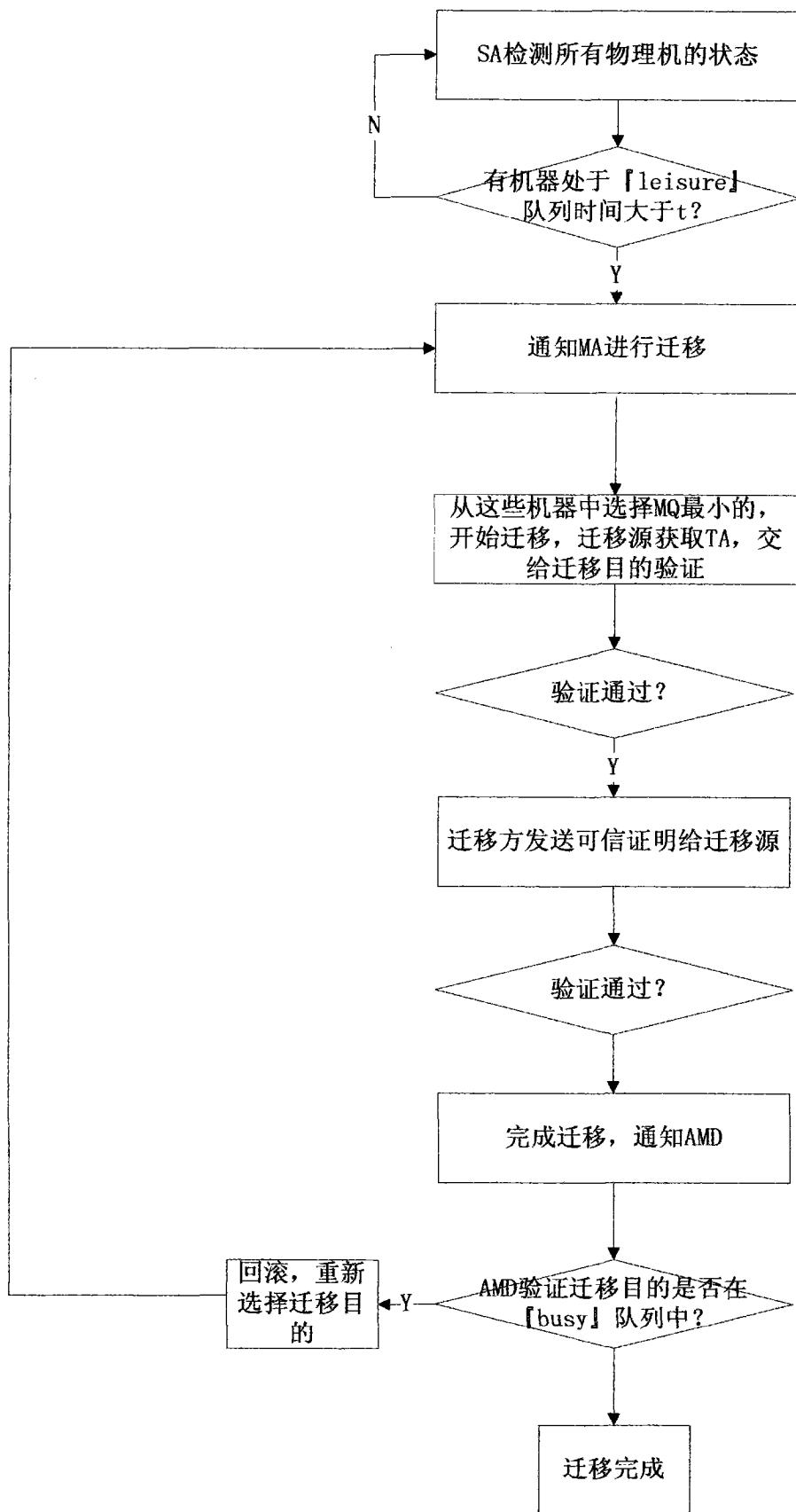


图 3