



(12) 发明专利申请

(10) 申请公布号 CN 103457775 A

(43) 申请公布日 2013. 12. 18

(21) 申请号 201310398784. 3

G06F 9/455(2006. 01)

(22) 申请日 2013. 09. 05

(71) 申请人 中国科学院软件研究所

地址 100190 北京市海淀区中关村南四街 4 号

(72) 发明人 黄涛 张文博 钟华 罗涛 吴恒 徐继伟

(74) 专利代理机构 北京科迪生专利代理有限公司 11251

代理人 成金玉 顾炜

(51) Int. Cl.

H04L 12/24(2006. 01)

H04L 12/26(2006. 01)

H04L 12/713(2013. 01)

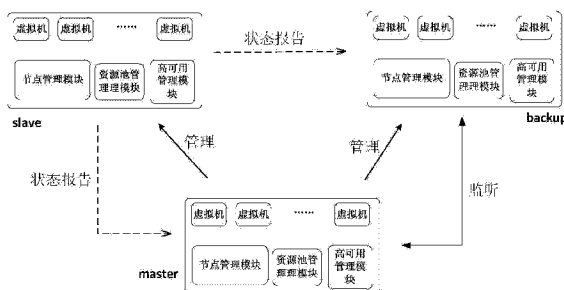
权利要求书2页 说明书8页 附图7页

(54) 发明名称

一种基于角色的高可用虚拟机池化管理系统

(57) 摘要

一种基于角色的高可用虚拟机池化管理系统,所述虚拟机池化管理方法中包括主节点即 master 节点、从节点即 slave 节点和备份节点即 backup 节点;master 节点是虚拟机资源池的唯一逻辑入口,负责管理资源池中的所有节点;slave 节点主要负责本节点的虚拟机生命周期的管理;backup 节点用于备份 master 节点的状态信息,保证与 master 节点的状态信息一致,并在 master 节点失效时接管其工作;所述虚拟机池化管理过程中各个节点会随着不同条件在三种角色,即 master 节点、从节点 slave 和备份节点 backup 中转换;通过序列法保障系统中只有一个 master 节点和备份节点,以及通过双向异步通信机制来保障 master 节点和备份节点的数据一致性。本发明提高了虚拟化系统的可靠性以及管理的灵活性。



1. 一种基于角色的高可用虚拟机池化管理系统,其特征在于:所述虚拟机池化中包括主节点即 master 节点、从节点即 slave 节点和备份节点即 backup 节点;master 节点是虚拟机资源池的唯一逻辑入口,负责管理资源池中的所有节点;slave 节点主要负责本节点的虚拟机生命周期的管理;backup 节点用于备份 master 节点的状态信息,保证与 master 节点的状态信息一致,并在 master 节点失效时接管其工作;所述虚拟机池化中各个节点所处的角色并不是固定的,而是随着不同条件在三种角色,即 master 节点、从节点 slave 和备份节点 backup 中转换;

所述每个节点均包括节点管理模块、资源池管理模块和高可用管理模块;每个节点均包括若干个虚拟机;

节点管理模块:负责本节点的虚拟机的生命周期管理,包括虚拟机的创建、关闭、启动、迁移,周期性地监听本节点和运行其上的每个虚拟机的资源使用情况,并写入本地的文件方便远程客户端解析;周期性地向 master 节点和 backup 节点发送状态信息,状态信息包括运行了哪些虚拟机,虚拟机关联的磁盘文件;同时还接收来自 master 节点的操作请求,根据这个转发的请求在本节点完成相应的操作;

资源池管理模块:当节点作为 master 时,该模块功能开启,负责管理虚拟机资源池中的每个节点即 slave 和 backup 节点,接受来自 slave 和 backup 节点的状态信息,判定它们是否有效;同时负责接收用户的请求,将请求转发给正确的目标节点;

高可用管理模块:当节点为 master 节点和 backup 节点时,该模块功能开启,该模块通过基于角色的双机热备方法,保障在 master 节点宕机时,系统能迅速恢复功能服务;通过序列法来保障资源池中只有一个 master 节点和 backup 节点,利用双向异步通信机制来保障 master 节点和 backup 节点两者状态信息的一致性;

上述模块中,资源池管理模块和高可用管理模块共同负责管理各个节点在三种角色中的转换。

2. 根据权利要求 1 所述的一种基于角色的高可用虚拟机池化管理系统,其特征在于:所述虚拟机池化中各个节点在三种角色,即 master 节点、从节点 slave 和备份节点 backup 中转换过程如下:

用三元组 $P=\langle \text{Number}, \text{Role}, \text{Sequence} \rangle$ 来描述每个节点,其中不同的元素的具体含义如下:Number, 即节点的编号,虚拟机池化中一共有三个节点,用 N1, N2 和 N3 来分别表示 master, backup 和 slave 这三个节点;Role, 即节点的角色, master, backup 和 slave 三种角色,每种角色执行不同功能;Sequence, 即节点的时间序号,每个节点都会有一个时间序号,实现步骤如下:

(1) 用户建立虚拟机池化时指定一个 none 节点作为 master 节点;

(2) 当用户需要将一个 none 节点加入虚拟机池化时,该节点从 none 转变为 slave 节点,成为虚拟机池化中的成员节点;

(3) 用户将某个节点从虚拟机池化中移除后,该节点由 slave 节点转化为 none 节点,不再属于这个虚拟机池化;

(4) 当虚拟机池化中的 backup 节点未被选取或者之前选取的 backup 节点失效时,由 master 节点选举出一个有效的 slave 节点,使其成为 backup 节点;

(5) 当 backup 节点探测到虚拟机池化中的 master 节点失效时,主动转换为 master,接

管 master 节点的工作；

(6) 当 backup 节点故障恢复后,重新回到虚拟机池化中,转变为 slave 节点；

(7) 当 master 节点故障恢复后,重新回到虚拟机池化中,并转变为 slave 节点。

3. 根据权利要求 1 所述的一种基于角色的高可用虚拟机池化管理系统,其特征在于:所述基于角色的双机热备方法具体如下:其中定义 S_i 为虚拟机池中的第 i 个节点, $1 \leq i \leq n$; 如果节点的角色为 master 时,具体执行步骤如下:

(1) master 节点监听 backup 节点是否有效或者未选举,如果 backup 节点有效则转步骤(1)继续监听这个节点,如果无效则转步骤(2);

(2) master 节点遍历资源池中的所有节点,若发现一个有效的节点,将这个节点角色转换为 backup,并将保存的状态信息备份到这个节点上,转步骤(1)监听这个 backup 节点,否则这次选举失败转步骤(2)继续进行选举;

如果节点的角色为 backup 时,具体执行步骤如下:

(1) backup 节点监听资源池中的 master 节点是否有效,如果有效则转步骤(1)继续进行监听,如果无效则转步骤(2);

(2) backup 节点将自身角色转换为 master 节点,并开启对应 master 节点的服务,同时通知资源池中其他节点 master 节点发生了变化。

4. 根据权利要求 1 所述的一种基于角色的高可用虚拟机池化管理系统,其特征在于:所述序列法具体实现如下:

(1) slave 节点的序列号为 0;

(2) 当每个节点被重新初始化时序列号都为 0;

其中定义 $Sequence(i)$ 表示第 i 个节点的序列号,以 master 节点表示为 $\langle N_i, master, k \rangle$ 为例,它选举出第 j 个有效节点为 backup 节点,将这个节点的序列号设置为 $k+1$,此时这个 j 节点表示为 $\langle N_j, backup, k+1 \rangle$; Master 节点即 $\langle N_i, master, k \rangle$ 每隔一段时间会检查资源池中的每个节点,若发现某个节点的角色不为 slave,并且序列号小于等于自身,则通知这个节点关闭角色对应的服务,转换为 slave 节点,节点 $\langle N_i, master, k \rangle$ 只会把角色不为 slave,并且序列号小于等于 k 的节点转换为 slave 节点,并不会影响到由它选举出来的 $\langle N_j, backup, k+1 \rangle$ 备份节点,通过序列法能够保障资源池中只有一个 master 节点和 backup 节点。

5. 根据权利要求 1 所述的一种基于角色的高可用虚拟机池化管理系统,其特征在于:所述双向异步通信机制具体实现如下:来自客户端的操作请求发送给 master 节点,由它转发给 slave 节点,slave 节点完成操作后发送响应消息给 master,由 master 将响应消息发送给客户端。slave 节点每隔一段时间将自身的状态信息发送给 master 和 backup 节点, master 和 backup 节点收到来自 slave 的状态信息,则更新保存的状态信息,这样即使在 master 节点宕机时,slave 节点也能将本节点的状态信息及时通知给新的 master 节点;同时 master 节点每隔一段时间广播一个消息给资源池中的所有节点,使得每个节点都能知道 master 和 backup 节点的目标地址,避免某个节点宕机恢复的过程中 master 节点和 backup 节点的地址发生变化时,不能将状态信息发送到正确的目标地址。

一种基于角色的高可用虚拟机池化管理系统

技术领域

[0001] 本发明涉及一种基于角色的高可用虚拟机池化管理系统,该系统扩展了传统的故障切换集群高可用保障技术,用于解决虚拟机池化管理的单点失效导致的故障恢复时间长和管理复杂的问题,属于软件技术领域。

背景技术

[0002] 虚拟机能够为操作系统和应用程序提供一个虚拟的计算机系统,它所构造的运行环境能够运行一个完整的操作系统,对上层的应用程序完全透明。虚拟机池化管理是指以统一管理视角提供虚拟机生命周期管理和资源按需提供的虚拟化技术。近几年来,虚拟机池化管理已成为构建主流云计算平台的关键技术之一。

[0003] 虚拟机池化管理普遍采取 master/slave 的管理模式,通过统一的逻辑入口来管理整个资源池中节点(本发明中的节点指的是物理服务器)及运行其上的虚拟机。代表的产品有 Microsoft Hyper-V, Citrix XenServer 和 VMware vSphere 等。通过虚拟机池化管理带来以下几个优点:(1) 虚拟机对节点的资源使用率是动态变化的,虚拟机资源池中的节点作为硬件资源提供给虚拟机使用。当某个时刻某个节点上的虚拟机消耗资源相对过高时,可以通过虚拟机在线迁移技术将它迁移到其它资源相对充裕的节点上,整个迁移过程对用户透明,保障资源的合理分配;(2) 虚拟机池化管理通常采用“共享存储”的模式。所谓“共享存储”,是指虚拟机资源池中所有虚拟机磁盘文件会统一存储在磁盘阵列或其它存储设备上,各节点仅能通过网络连接存储设备实现虚拟机的实例化(内存态)。在这种管理模式下,当节点宕机导致运行其上的虚拟机不可用时,容易通过虚拟机实例在其它节点重启的模式实现虚拟机的高可用。

[0004] 虚拟机池化管理带了管理简单优点的同时,也引入了风险集中,单点失效问题。在 master/slave 管理模式下, master 节点保存了虚拟机资源池所有节点的全局状态信息,因此 master 节点宕机可能导致整个虚拟机资源池不可用或虚拟机资源池重装等严重后果。单点失效问题涉及到系统的高可用,高可用保障方法可以分为以下四大类 (Chan H, Chieu T. An approach to high availability for cloud servers with snapshot mechanism[C]//Proceedings of the Industrial Track of the 13th ACM/IFIP/USENIX International Middleware Conference. ACM, 2012:6.): 镜像技术 (Mirroring)、复制技术 (Replication)、故障切换集群 (Failover clustering) 和快照技术 (snapshot)。

[0005] 镜像技术是指设置主节点和镜像节点,镜像节点每隔一段时间从主节点主动取数据备份到自身。复制技术是基于订阅者的模式,主节点在自身状态信息发生变化时主动发送给所有订阅的备份节点,通知其更新状态信息。故障切换集群通过多个节点组成一个集群,集群中任何一个节点发生故障都可以由其他节点接管。快照技术是通过节点每隔一段时间备份一个还原点,当节点发生故障后可以通过这个备份的还原点来恢复系统。

[0006] 镜像技术和复制技术被 HDFS 文件系统 (Borthakur D, Gray J, Sarma J S, et al. Apache Hadoop goes realtime at Facebook[C]//Proceedings of

the2011international conference on Management of data. ACM, 2011:1071-1080.) 用于解决 NameNode 节点的单点失效问题。通过设置一个备份节点,当用户对文件系统进行写操作导致文件系统的目录结构发生改变时,主 NameNode 通过复制技术将这一操作请求写入一个 NFS 共享节点上,然后备份的节点通过镜像技术不断的读取 NFS 共享节点上的操作请求执行,保障与主 NameNode 节点上保存的信息一致,在主节点失效时能接管其工作,但是这种方式备份节点是静态指定,缺乏灵活性,并且 NFS 节点也是个单点问题。

[0007] 故障切换集群被广泛应用与 XenServer 和 vSphere 虚拟机池化管理系统中,通过 slave 节点来探知 master 节点是否失效,当某个 slave 节点探知到 master 节点失效时,通过一定的选举算法,比如 Bully 算法 (Garcia-Molina H. Elections in a distributed computing system[J]. Computers, IEEE Transactions on, 1982, 100 (1):48-59) 选举出一个新的节点充当 master 节点恢复之前保存的数据,继续管理资源池中的成员节点和虚拟机。由于新的 master 节点是通过所有的 slave 节点在故障发生后根据选举算法协同选举出来的,所以恢复时间相对较长。

[0008] 此外的快照技术 (snapshot),如果在设置的还原点之前需要备份数据改变并且发生故障,系统将难以恢复到故障发生之前的状态。

[0009] 综上所述,上述几种方法中存在的缺乏灵活性和故障恢复时间长的缺点。

发明内容

[0010] 本发明技术解决问题:克服现有技术的不足,提供一种基于角色的虚拟机池高可用系统,提高了系统的灵活性和减少故障的恢复时间。

[0011] 本发明技术解决方案:一种基于角色的高可用虚拟机池化管理系统,通过序列列表保障系统中只有一个 master 节点和备份节点,以及通过双向异步通信机制来保障 master 节点和备份节点的数据一致性。

[0012] 通过增加一个 backup 节点的角色,这个节点由 master 节点主动选举出来,然后 master 节点失效由 backup 节点主动接管其工作,这样达到 backup 节点自动选取和 master 节点主动替换的效果。如图 1 所示,该系统包括三个模块:

[0013] 1) 节点管理模块:负责本节点的虚拟机的生命周期管理,包括虚拟机的创建,关闭,启动,迁移,周期性地监听本节点和运行其上的每个虚拟机的资源使用情况,并写入本地的文件方便远程客户端解析;周期性地向 master 节点和 backup 节点发送状态信息,状态信息包括运行了哪些虚拟机,虚拟机关联的磁盘文件等;

[0014] 2) 资源池管理模块:负责管理虚拟机资源池中的每个节点,如监听资源池中每个节点是否有效。根据用户的请求转发给目标节点进行相应的操作。提供虚拟机资源池的建立和弹性资源供给,即节点的添加和退出;

[0015] 3) 高可用管理模块:在 master 节点上,高可用模块负载监听 backup 节点的是否有效,当无效时,选举出新的备份节点继续监听。在 backup 节点上,高可用模块负载监听 master 节点是否有效,当无效时主动接管其工作,并广播给虚拟机资源池中的所有节点这一变化。master 节点和 backup 节点同时负责接收来自 slave 节点的心跳信息,根据心跳信息来更新自身保存的状态信息。

[0016] 在基于角色的双机热备高可用系统中,主要涉及到两种方法和一种机制,分别是

基于角色的双机热备方法、序列法和双向异步通信机制。通过基于角色的双机热备方法消除 master 节点单点失效问题带来的服务中断。通过序列法保障资源池中只有一个 master 节点提供与用户交互的入口。通过双向异步通信机制保障 master 节点和 backup 节点两者的状态信息的一致性。

[0017] 以下是详细描述。

[0018] 3.1 基于角色的节点描述和转换关系,如图 2 所示。

[0019] 为了方便后面的表述,本文用三元组 $P=\langle \text{Number}, \text{Role}, \text{Sequence} \rangle$ 来描述每个节点,其中不同的元素的具体含义如下:

[0020] 1) Number, 即节点的编号,如资源池中一共有三个节点,用 N1, N2 和 N3 来分别表示这三个节点;

[0021] 2) Role, 即节点的角色,这里有 master, backup 和 slave 三种角色,每种角色执行不同功能;

[0022] 3) Sequence, 即节点的时间序号,每个节点都会有一个时间序号。

[0023] 例如, $\langle N3, \text{backup}, 35 \rangle$ 表示第 3 个节点的角色为 backup, 并且它的时间序号为 35。

[0024] 虚拟机资源池中的每个节点都赋予了一定的角色,每个角色具有不同的功能。分别如下:

[0025] 1) master 节点也称为主节点,它是虚拟机资源池的唯一逻辑入口,负责管理资源池中的所有节点;

[0026] 2) slave 节点主要负责本节点的虚拟机生命周期的管理;

[0027] 3) backup 节点用于备份 master 节点的状态信息,保证与 master 节点的状态信息一致,并在 master 节点失效时接管其工作。

[0028] 资源池中各个节点所处的角色并不是固定的,而是随着不同条件在三种角色中转换,转换关系如图 2 所示,其中 none 表示节点不赋予任何角色,不在虚拟机资源池中:

[0029] (1) 用户建立虚拟机资源池时指定一个 none 节点作为 master 节点;

[0030] (2) 当用户需要将一个 none 节点加入虚拟机资源池时,该节点从 none 转变为 slave 节点,成为资源池中的成员节点;

[0031] (3) 用户将某个节点从虚拟机资源池中移除后,该节点由 slave 节点转化为 none 节点,不再属于这个虚拟机资源池;

[0032] (4) 当虚拟机资源池中的 backup 节点未被选取或者之前选取的 backup 节点失效时,由 master 节点选举出一个有效的 slave 节点,使其成为 backup 节点;

[0033] (5) 当 backup 节点探测到虚拟机资源池中的 master 节点失效时,主动转换为 master,接管 master 节点的工作;

[0034] (6) 当 backup 节点故障恢复后,重新回到资源池中,转变为 slave 节点。

[0035] (7) 当 master 节点故障恢复后,重新回到虚拟机资源池中,并转变为 slave 节点;

[0036] 3.2 基于角色的双机热备方法

[0037] master/slave 模式下要消除单点失效问题,关键是在 master 节点发生故障时,能够有一个节点被选举出来主动接管其工作。本发明实现的基于角色的双机热备方法,该方法实现在 master 节点和 backup 节点的高可用模块,具体的算法如图 3 所示,其中定义 $S_i (1 \leq i \leq n)$ 为虚拟机池中的第 i 个节点。

[0038] 如果节点的角色为 master 时,具体执行步骤如下:

[0039] 1) master 节点监听 backup 节点是否有效或者未选举,如果 backup 节点有效则转步骤 1) 继续监听这个节点,如果无效则转步骤 2);

[0040] 2) master 节点遍历资源池中的所有节点,若发现一个有效的节点,将这个节点角色转换为 backup,并将保存的状态信息备份到这个节点上,转步骤 1) 监听这个 backup 节点。否则这次选举失败转步骤 2) 继续进行选举。

[0041] 如果节点的角色为 backup 时,具体执行步骤如下:

[0042] 1) backup 节点监听资源池中的 master 节点是否有效,如果有效则转步骤 1) 继续进行监听,如果无效则转步骤 2);

[0043] 2) backup 节点将自身角色转换为 master 节点,并开启对应 master 节点的服务,同时通知资源池中其他节点 master 节点发生了变化。

[0044] 3.3 序列法

[0045] 在基于角色的双机热备方法中, master 节点和 backup 节点的失效判定是通过一定的时间内能否接收到对方的响应消息来判定的。如果节点连接的网络短暂不可用,或者节点过于繁忙导致不能及时响应请求,则会被判定为发生宕机现象。这样会导致资源池中出现多个 master 节点和 backup 节点的情况,造成多个 master 和 backup 节点之间保存的数据不一致的现象。

[0046] 传统的解决方法是通过仲裁的方式,即设置一个参考的 IP,当 master 节点连接的网络发生故障时, master 节点 ping 参考 IP 失败,则转换为 slave 节点,而 backup 节点 ping 参考 IP 成功,则转换为 master 节点选取出新的 backup 节点。当 backup 节点连接的网络发生故障时, master 节点 ping 参考 IP 成功,则选取出新的 backup 节点,而 backup 节点 ping 参考 IP 失败,则转换为 slave 节点。但是这种方法不能解决节点繁忙的情况,节点繁忙时 ping 命令也不能及时执行。

[0047] 为了解决这个问题,保障上述的方法有效,发明了序列法。具体的规定如下:

[0048] (1) slave 节点的序列号为 0;

[0049] (2) 当每个节点被重新初始化时序列号都为 0。

[0050] 该方法实现于 master 节点的高可用模块,具体的算法如图 4 所示,其中定义 Sequence(i) 表示第 i 个节点的序列号。以 master 节点表示为 $\langle N_i, \text{master}, k \rangle$ 为例,它选举出第 j 个有效节点为 backup 节点,将这个节点的序列号设置为 k+1,此时这个 j 节点表示为 $\langle N_j, \text{backup}, k+1 \rangle$ 。Master 节点即 $\langle N_i, \text{master}, k \rangle$ 每隔一段时间会检查资源池中的每个节点,若发现某个节点的角色不为 slave,并且序列号小于等于自身,则通知这个节点关闭角色对应的服务,转换为 slave 节点。可以看到节点 $\langle N_i, \text{master}, k \rangle$ 只会把角色不为 slave,并且序列号小于等于 k 的节点转换为 slave 节点,并不会影响到由它选举出来的 $\langle N_j, \text{backup}, k+1 \rangle$ 备份节点。通过序列法能够保障资源池中只有一个 master 节点和 backup 节点。

[0051] 3.4 双向异步通信机制,如图 10 所示

[0052] 系统采用基于角色的双机热备的方法来保障 master 节点的高可用,最重要的是保证 backup 节点和 master 节点保存的状态信息的一致性,这样 backup 节点才能无故障的接管 master 节点的工作。

[0053] 为了保证这两个节点的状态信息的一致性,发明了一种双向异步通信机制,来自客户端的操作请求发送给 master 节点,由它转发给 slave 节点,slave 节点完成操作后发送响应消息给 master,由 master 将响应消息发送给客户端。slave 节点每隔一段时间将自身的状态信息发送给 master 和 backup 节点, master 和 backup 节点收到来自 slave 的状态信息,则更新保存的状态信息。这样,即使在 master 节点宕机时,slave 节点也能将本节点的状态信息及时通知给新的 master 节点。同时 master 节点每隔一段时间广播一个消息给资源池中的所有节点,使得每个节点都能知道 master 和 backup 节点的目标地址,避免某个节点宕机恢复的过程中 master 节点和 backup 节点的地址发生变化时,不能将状态信息发送到正确的目标地址。

[0054] 与现有技术相比,本发明具有如下技术优势:

[0055] (1)本发明中的 master 节点的故障判断和接管,backup 节点的故障检测和主动选取都是有系统自身完成,无需人工干预,能够有效地解决单点失效问题。同时由于 backup 节点是由 master 节点在正常提供服务的过程中选举出来的,相比传统方法 master 节点出现故障后由其他节点探知然后协调选举出一个节点接管,本发明中的方法故障恢复时间短。

[0056] (2)本发明中的序列法,能够保障资源池中只存在一个 master 节点和 backup 节点。相比现有的技术,能够处理由于节点服务繁忙带来的“假死”现象。

[0057] (3)本发明才用了双向异步通信机制来保证 master 节点和 backup 节点的数据一致性,能在通过 slave 节点发送的状态信息,来恢复整个资源池的全局状态信息。

[0058] (4)上述几种方法中存在的缺乏灵活性和故障恢复时间长的缺点。本发明基于故障切换集群技术,实现了一种基于角色的双机热备的高可用系统。该系统通过备份节点的自动选举和 master 节点故障的主动接管替换的方式提高了系统的灵活性和减少故障的恢复时间。通过序列法保障系统中只有一个 master 节点和备份节点,以及通过双向异步通信机制来保障 master 节点和备份节点的数据一致性。

[0059] (5)在基于角色的双机热备方法中, master 节点和 backup 节点的失效判定是通过一定的时间内能否接收到对方的响应消息来判定的。如果节点连接的网络短暂不可用,或者节点过于繁忙导致不能及时响应请求,则会被判定为发生宕机现象。这样会导致资源池中出现多个 master 节点和 backup 节点的情况,造成多个 master 和 backup 节点之间保存的数据不一致的现象。

[0060] (6)传统的解决方法是通过仲裁的方式,即设置一个参考的 IP,当 master 节点连接的网络发生故障时, master 节点 ping 参考 IP 失败,则转换为 slave 节点,而 backup 节点 ping 参考 IP 成功,则转换为 master 节点选取出新的 backup 节点。当 backup 节点连接的网络发生故障时, master 节点 ping 参考 IP 成功,则选取出新的 backup 节点,而 backup 节点 ping 参考 IP 失败,则转换为 slave 节点。但是这种方法不能解决节点繁忙的情况,节点繁忙时 ping 命令也不能及时执行。为了解决这个问题,发明了序列法。

附图说明

[0061] 图 1 为本发明系统的组成框图;

[0062] 图 2 为本发明中角色转换图;

- [0063] 图 3 为本发明中本基于角色的节点选举替换算法；
- [0064] 图 4 为本发明中序列法算法；
- [0065] 图 5 为本发明中序列法处理 master 节点网络故障示例图；
- [0066] 图 6 为本发明中序列法处理 master 节点“假死”示例图；
- [0067] 图 7 为本发明中序列法处理 backup 节点网络故障示例图；
- [0068] 图 8 为本发明中序列法处理 backup 节点“假死”示例图；
- [0069] 图 9 为本发明中的基于角色的双机热备的处理过程图；
- [0070] 图 10 为本发明中双向异步通信机制的处理过程图。

具体实施方式

[0071] 本发明基于角色的虚拟机池高可用保障技术及系统,通过 master 和 backup 节点的故障检测,故障恢复,能够有效地解决单点失效问题。以下结合具体实施例和附图对本发明进行详细说明。

[0072] 在系统的高可用模块中,通过基于角色的双机热备的处理方法,来保障资源池中 master 节点宕机时,有一个有效的 backup 节点接管它的工作,保障系统能够正常提供服务,以下是根据具体的场景实例进行阐述。

[0073] 1. 基于角色的双机热备的处理过程,如图 3 所示。

[0074] 在 3.1 节定义三元组的基础上,改变第三元来,用来表示节点是否有效(invalid 表示无效, valid 表示有效),假设资源池中有 5 个节点,分别表示为 $\langle N_1, \text{master}, \text{valid} \rangle$, $\langle N_2, \text{backup}, \text{valid} \rangle$, $\langle N_3, \text{slave}, \text{invalid} \rangle$, $\langle N_4, \text{slave}, \text{invalid} \rangle$ 和 $\langle N_5, \text{slave}, \text{valid} \rangle$, 如图 9(a) 所示。

[0075] (1)当 N_2 节点出现故障时,即表示为 $\langle N_2, \text{backup}, \text{invalid} \rangle$, N_1 节点探知其无效,开始遍历资源池中的节点,选举出有效节点；

[0076] (2)如图 9(b) 所示,遍历到 $\langle N_3, \text{slave}, \text{invalid} \rangle$ 时,发现其无效继续遍历。当遍历到 $\langle N_4, \text{slave}, \text{invalid} \rangle$ 时,发现其无效继续遍历。当遍历到 $\langle N_5, \text{slave}, \text{valid} \rangle$ 时,发现其有效,则将状态信息备份到这个节点上,并将其角色转换为 backup,结束此次选举,此时 N_5 节点表示为 $\langle N_5, \text{backup}, \text{valid} \rangle$ ；

[0077] (3)如图 9(c) 所示,当 N_1 节点出现故障,即表示为 $\langle N_1, \text{master}, \text{invalid} \rangle$, N_5 节点探知其无效,则将自身转换为 master 的角色,即表示为 $\langle N_5, \text{master}, \text{valid} \rangle$ ；

[0078] (4) N_5 节点始遍历资源池中的节点,选举出有效节点。当遍历到 $\langle N_1, \text{master}, \text{invalid} \rangle$ 时,发现其无效继续遍历。当遍历到 $\langle N_2, \text{backup}, \text{invalid} \rangle$ 时,发现其无效继续遍历。当遍历到 $\langle N_3, \text{slave}, \text{invalid} \rangle$ 时,发现其无效继续遍历。当遍历到 $\langle N_4, \text{slave}, \text{invalid} \rangle$ 时,发现其无效继续遍历。 N_5 节点在这次遍历过程没有发现有效的节点,将进行新一轮的遍历；

[0079] (5)如图 9(d) 所示,若某个时刻 N_3 节点故障恢复成为有效节点,即表示为 $\langle N_3, \text{slave}, \text{valid} \rangle$ 。它将被 N_5 节点探知到,转换为 backup 节点,即为 $\langle N_3, \text{backup}, \text{valid} \rangle$, 则 N_5 节点完成选举；

[0080] (6)如图 9(e) 所示,当某个时刻 N_1 节点和 N_2 节点故障恢复,它们都将转换为 slave 节点,即分别表示为 $\langle N_1, \text{slave}, \text{valid} \rangle$ 和 $\langle N_2, \text{slave}, \text{valid} \rangle$ 。

[0081] 在系统的高可用模块中,通过序列法,来克服网络故障带来的资源池中出现多个 master 节点冲突的情况,保障资源池中只有一个 master 节点来管理整个资源池的节点和虚拟机。以下是根据具体的场景实例来详细阐述这个方法的处理过程。

[0082] 2. 序列法的处理过程,如图 4 所示。

[0083] 当 master 节点出现网络故障时如图 5 所示:

[0084] (1) 当 $\langle N_i, \text{master}, k \rangle$ 出现网络故障,由它选举出来的 $\langle N_j, \text{backup}, k+1 \rangle$ 得不到它的响应消息,同时 $\langle N_i, \text{master}, k \rangle$ 也不能选取出新的节点充当 backup 节点;

[0085] (2) $\langle N_j, \text{backup}, k+1 \rangle$ 而转变为 $\langle N_j, \text{master}, k+1 \rangle$, 然后选取出一个 $\langle N_k, \text{backup}, k+2 \rangle$;

[0086] (3) 若 $\langle N_i, \text{master}, k \rangle$ 的网络恢复,它会选取出 $\langle N_i, \text{backup}, k+1 \rangle$;

[0087] (4) 由于 $\langle N_j, \text{master}, k+1 \rangle$ 是序列号最大的 master 节点,它会通知 N_i 和 N_k 节点关闭相应服务,转换为 slave 节点;

[0088] 当 master 节点出现“假死”现象时如图 6 所示,与上述不同在步骤 3), 当 $\langle N_i, \text{master}, k \rangle$ “假死”现象消除时,单方向监听节点 b 是否有效。最终会在步骤 4) 中,用 $\langle N_j, \text{master}, k+1 \rangle$ 通知其转换为 slave 节点。

[0089] 当 backup 节点出现网络故障时如图 7 所示:

[0090] (1) 当 $\langle N_j, \text{backup}, k \rangle$ 出现网络故障时,不能得到 $\langle N_i, \text{master}, k-1 \rangle$ 响应消息,同时 $\langle N_i, \text{master}, k-1 \rangle$ 也不能得到 $\langle N_j, \text{backup}, k \rangle$ 的响应消息;

[0091] (2) 由于网络故障, $\langle N_j, \text{backup}, k \rangle$ 转换为 $\langle N_j, \text{master}, k \rangle$ 并且不能选取出新的节点作为 backup 节点。而 $\langle N_i, \text{master}, k-1 \rangle$ 会重新选举一个新节点 $\langle N_k, \text{backup}, k \rangle$;

[0092] (3) 当节点 N_j 网络恢复时,选举出一个新节点 $\langle N_i, \text{backup}, k+1 \rangle$;

[0093] (4) 由于 $\langle N_i, \text{master}, k \rangle$ 是序列号最大的 master 节点,它会通知 N_j 和 N_k 节点关闭相应服务,转换为 slave 节点。

[0094] 当 backup 节点出现“假死”现象时如图 8 所示:

[0095] (1) 当 $\langle N_j, \text{backup}, k \rangle$ 出现“假死”现象, $\langle N_i, \text{master}, k-1 \rangle$ 不能得到它的响应消息, $\langle N_j, \text{backup}, k \rangle$ 此刻也暂时停止监听 $\langle N_i, \text{master}, k-1 \rangle$;

[0096] (2) 由于 $\langle N_i, \text{master}, k-1 \rangle$ 不能得到 N_j 节点的响应消息,则选举出 $\langle N_k, \text{backup}, k \rangle$;

[0097] (3) N_j 节点的“假死”现象消除,它会单方向监听 N_i 节点;

[0098] (4) 当 N_i 节点出现宕机时, N_j 节点和 N_k 节点都能探知 N_i 发生故障;

[0099] (5) 由于得不到 N_i 节点的响应, N_k 节点转换为 $\langle N_k, \text{master}, k \rangle$, 选举出一个新节点 $\langle N_m, \text{backup}, k+1 \rangle$ 。同时 N_j 节点转换为 $\langle N_j, \text{master}, k \rangle$, 选举出一个新节点 $\langle N_n, \text{backup}, k+1 \rangle$ 。当 N_j 节点通知 N_k 节点转换为 slave 节点时, N_k 转换为 $\langle N_k, \text{slave}, 0 \rangle$, N_m 节点单方向监听 N_k 节点。当 N_k 节点出现宕机, N_m 节点会成为序列号最大的 master 节点,通知 N_j 和 N_n 节点转换为 slave 节点。当 N_k 节点通知 N_j 节点转换为 slave 节点的情况与上述类似。

[0100] 3. 双向异步通信机制的处理过程,如图 10 所示。

[0101] 定义 $VM(i, j)$ 表示第 i 个节点上的第 j 个虚拟机。假设资源池中有 5 个节点和 3 个虚拟机,这 5 个节点分别表示为 $\langle N_1, \text{master}, \text{valid} \rangle$, $\langle N_2, \text{backup}, \text{valid} \rangle$, $\langle N_3, \text{slave}, \text{valid} \rangle$, $\langle N_4, \text{slave}, \text{valid} \rangle$ 和 $\langle N_5, \text{slave}, \text{valid} \rangle$, 3 个虚拟机分别表示为 $VM(4, 1)$, $VM(4, 2)$ 和

VM(5, 1), 如图 10(a) 所示。

[0102] (1) 客户端通知 $\langle N_1, \text{master}, \text{valid} \rangle$ 节点, 要求将 N_4 节点上的第 1 个虚拟机迁移到 N_5 节点上;

[0103] (2) 如图 10(b) 所示, N_1 节点将操作请求转发给 N_4 节点, N_4 节点接收这个操作请求开始将 VM(4, 1) 虚拟机迁移到 N_5 节点;

[0104] (3) 如图 10(c) 所示, 在迁移的过程中, N_1 节点发生宕机故障, 即表示为 $\langle N_1, \text{master}, \text{invalid} \rangle$, 此时迁移过程还在进行, 而 N_2 节点成为 master 节点, 即为 $\langle N_2, \text{master}, \text{valid} \rangle$ 。同时 N_2 节点选举出 N_3 节点充当 backup 节点, 此时 N_3 节点表示为 $\langle N_3, \text{backup}, \text{valid} \rangle$;

[0105] (4) 此时迁移过程完成, VM(4, 1) 成为 VM(5, 2), N_4 和 N_5 节点将自身的状态信息通知给 N_1 和 N_2 节点。由于 N_1 节点宕机, 它将不接受来自 N_4 和 N_5 的状态信息。Master 节点 N_2 接收到来自 N_4 和 N_5 的状态信息进行更新, 则它保存的虚拟机的状态信息为 VM(4, 2), VM(5, 1) 和 VM(5, 2)。而 backup 节点 N_3 接收不到来自 N_4 和 N_5 的状态信息, 则它保存的虚拟机的状态信息还是为 VM(4, 1), VM(4, 2) 和 VM(5, 1);

[0106] (5) N_2 节点遍历资源池中的所有节点, 并将 master 节点 N_2 和 backup 节点 N_3 的位置信息通知给这些节点;

[0107] (6) N_4 节点和 N_5 节点接收到来自 master 节点 N_2 发送的位置信息, 则开始将自身的状态信息发送 N_2 和 N_3 节点;

[0108] (7) 如图 10(d) 所示, N_3 节点接收到来自 N_4 和 N_5 节点的状态信息进行更新, 则它保存的虚拟机的状态信息为 VM(4, 2), VM(5, 1) 和 VM(5, 2), 与 N_2 节点的状态信息一致;

[0109] (8) 如图 10(d) 所示, N_1 节点宕机恢复后将成为 slave 节点, 即表示为 $\langle N_1, \text{slave}, \text{valid} \rangle$, 此时它会将自身的状态信息发送给 N_1 和 N_2 节点。当 N_1 将接收到来自 master 节点 N_2 发送的位置信息, 开始将自身的状态信息发送给 N_2 和 N_3 节点。

[0110] 本发明未详细阐述部分属于本领域公知技术。

[0111] 以上所述, 仅为本发明部分具体实施方式, 但本发明的保护范围并不局限于此, 任何熟悉本领域的人员在本发明揭露的技术范围内, 可轻易想到的变化或替换, 都应涵盖在本发明的保护范围之内。

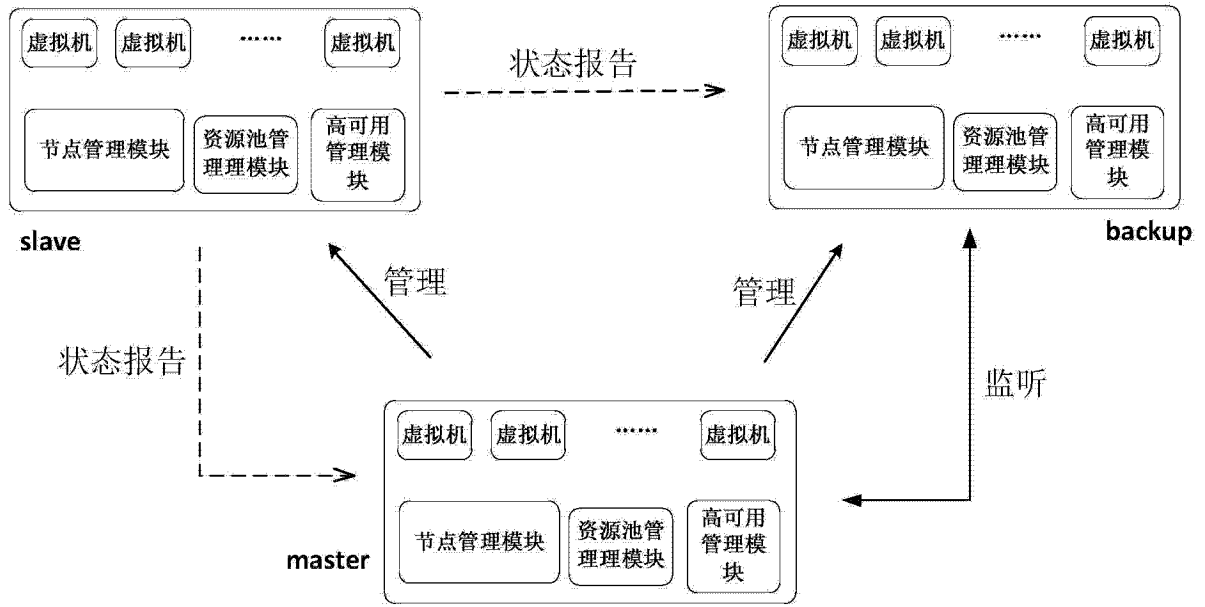


图 1

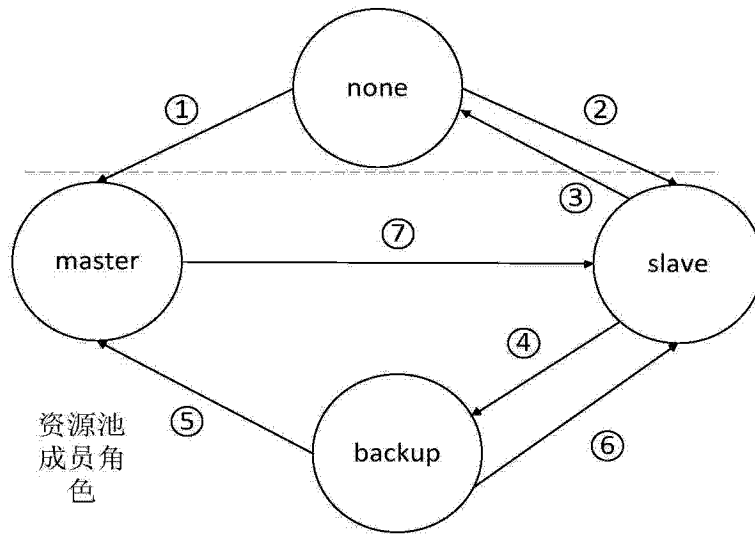


图 2

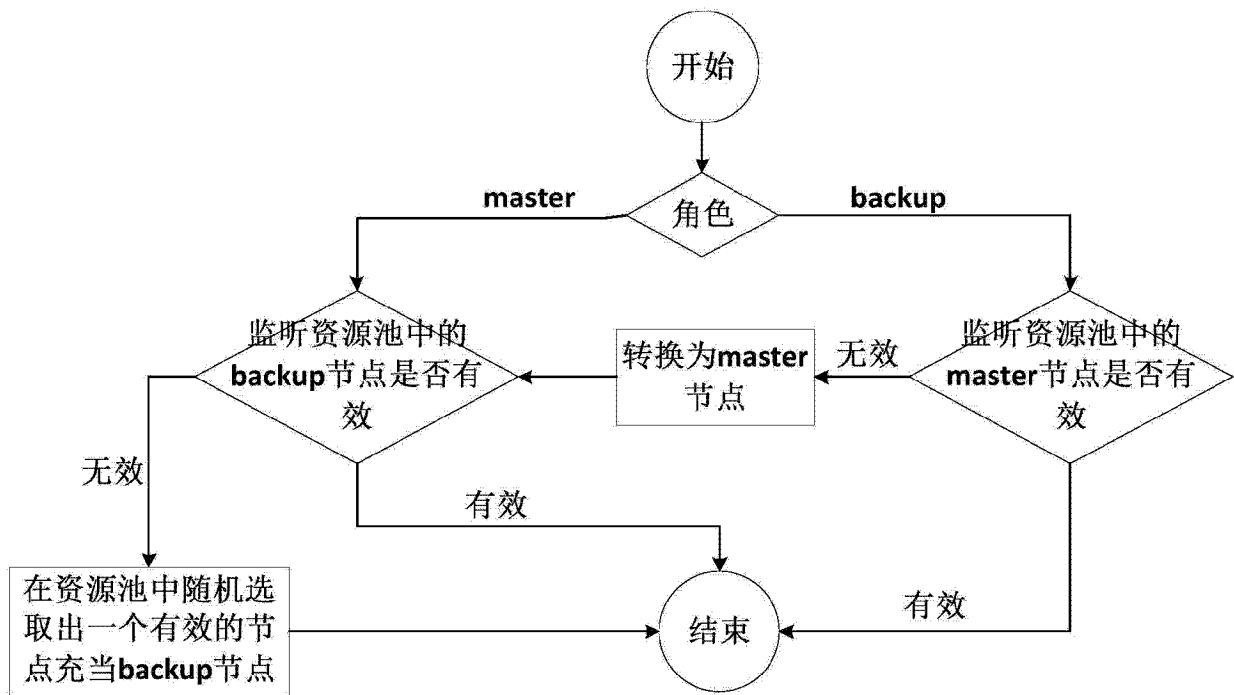


图 3

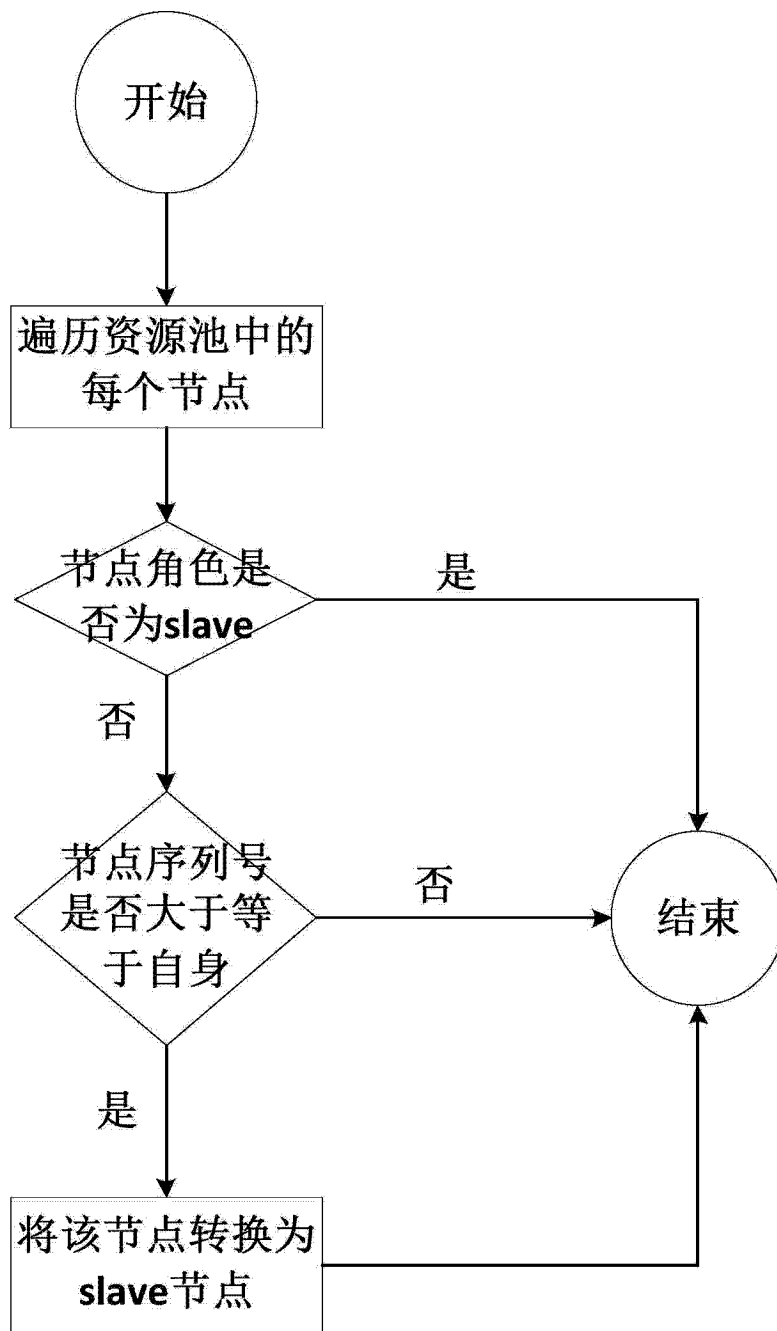


图 4

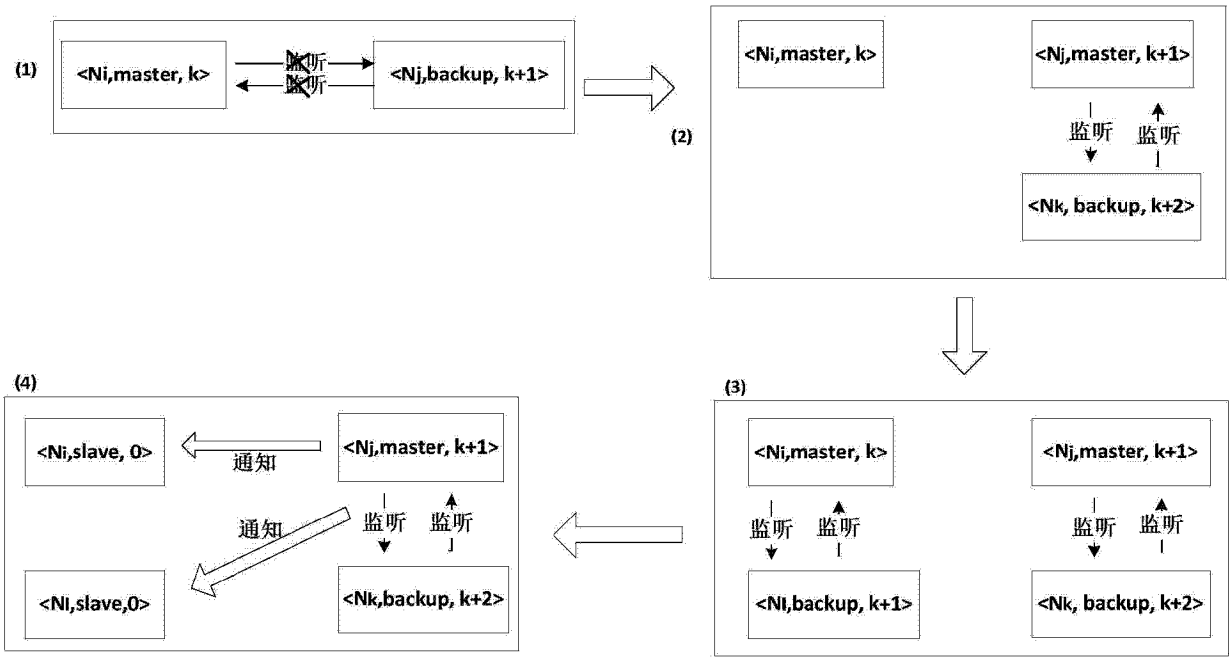


图 5

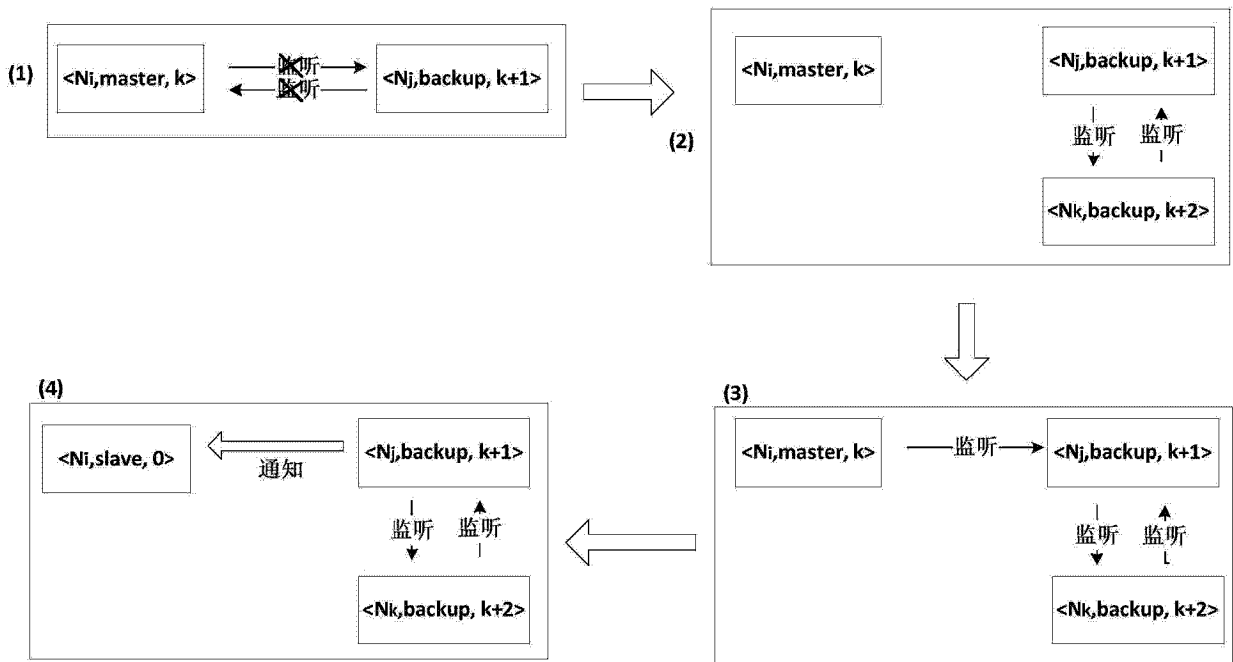


图 6

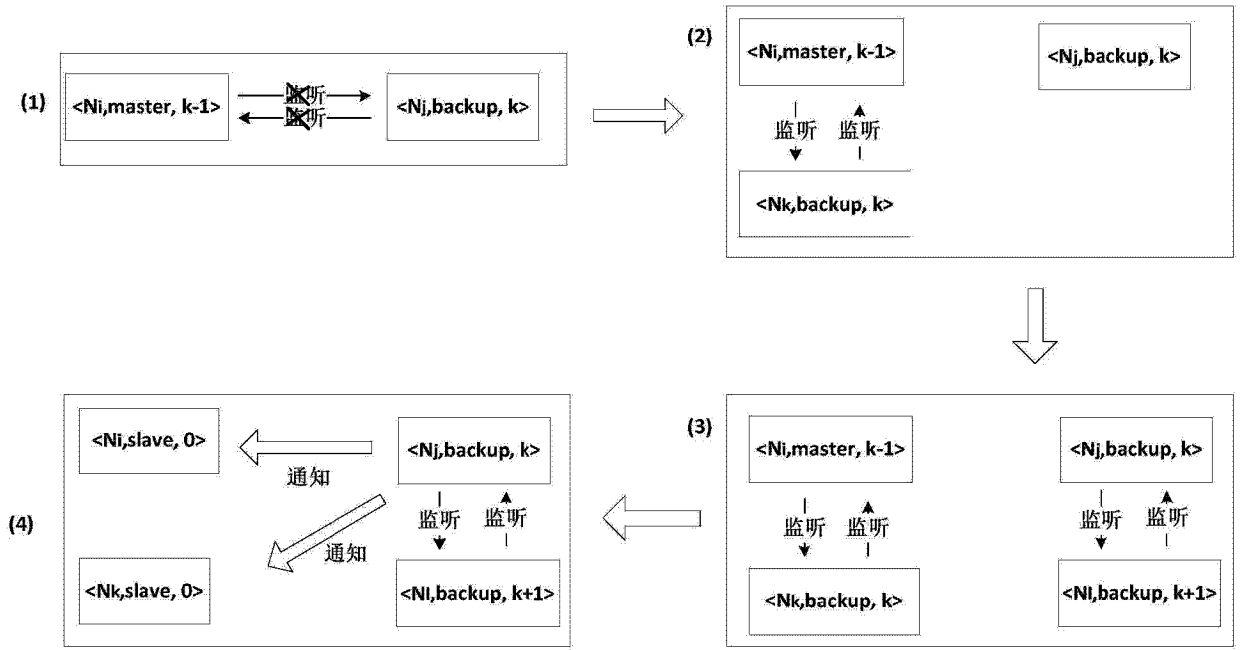


图 7

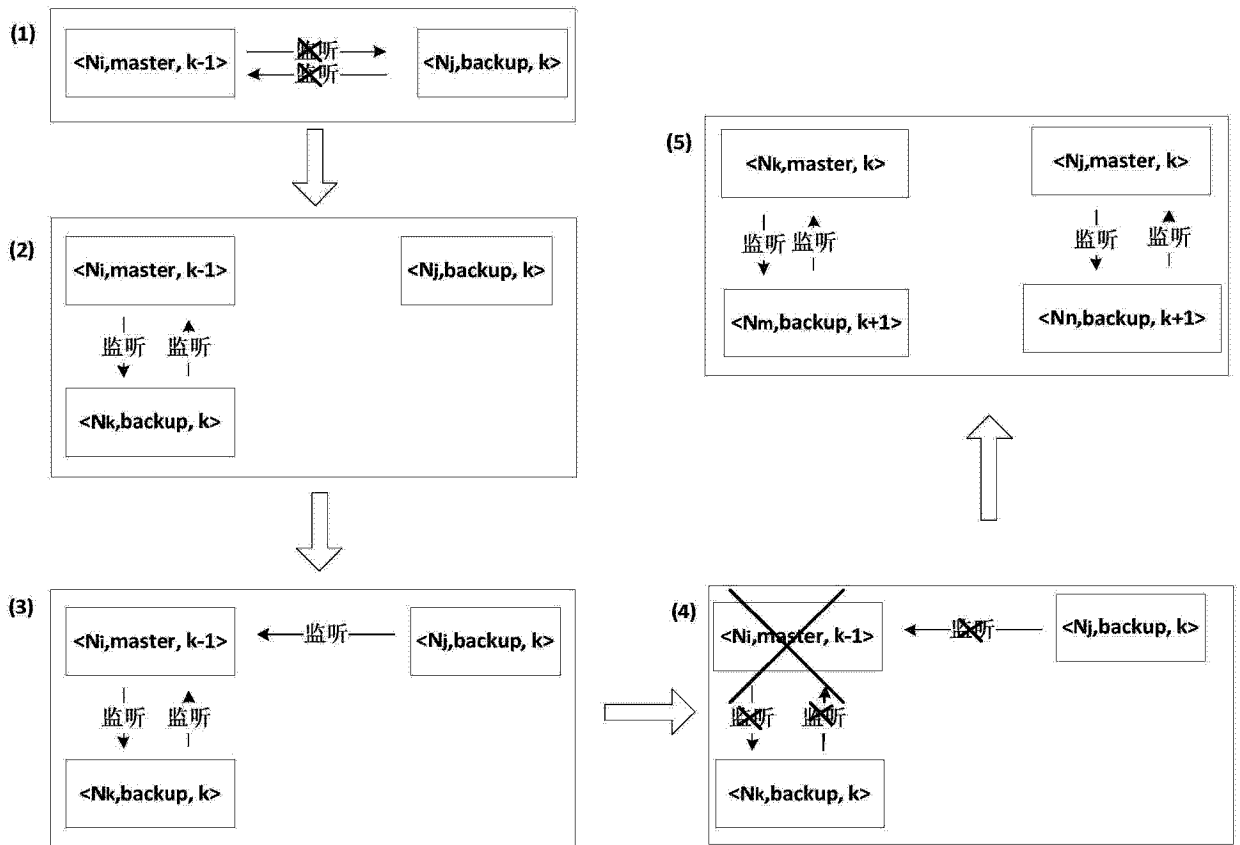


图 8

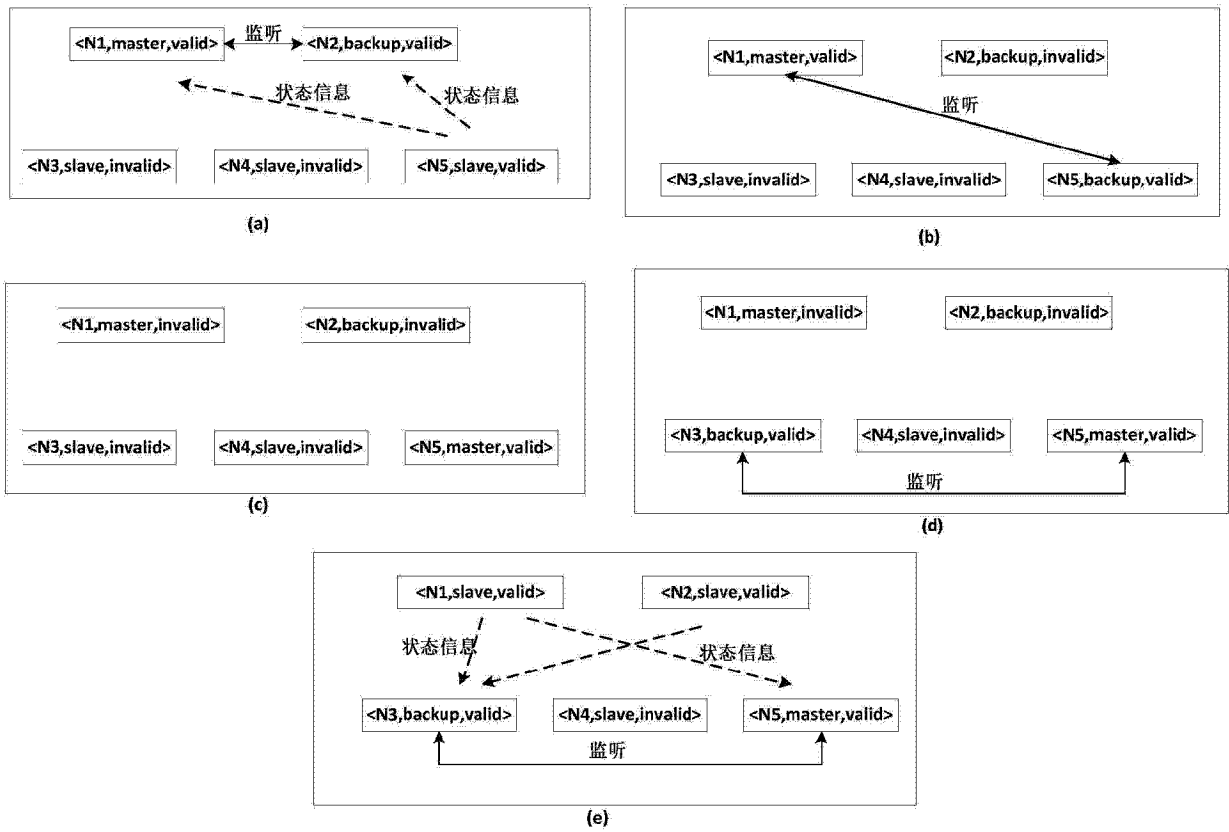


图 9

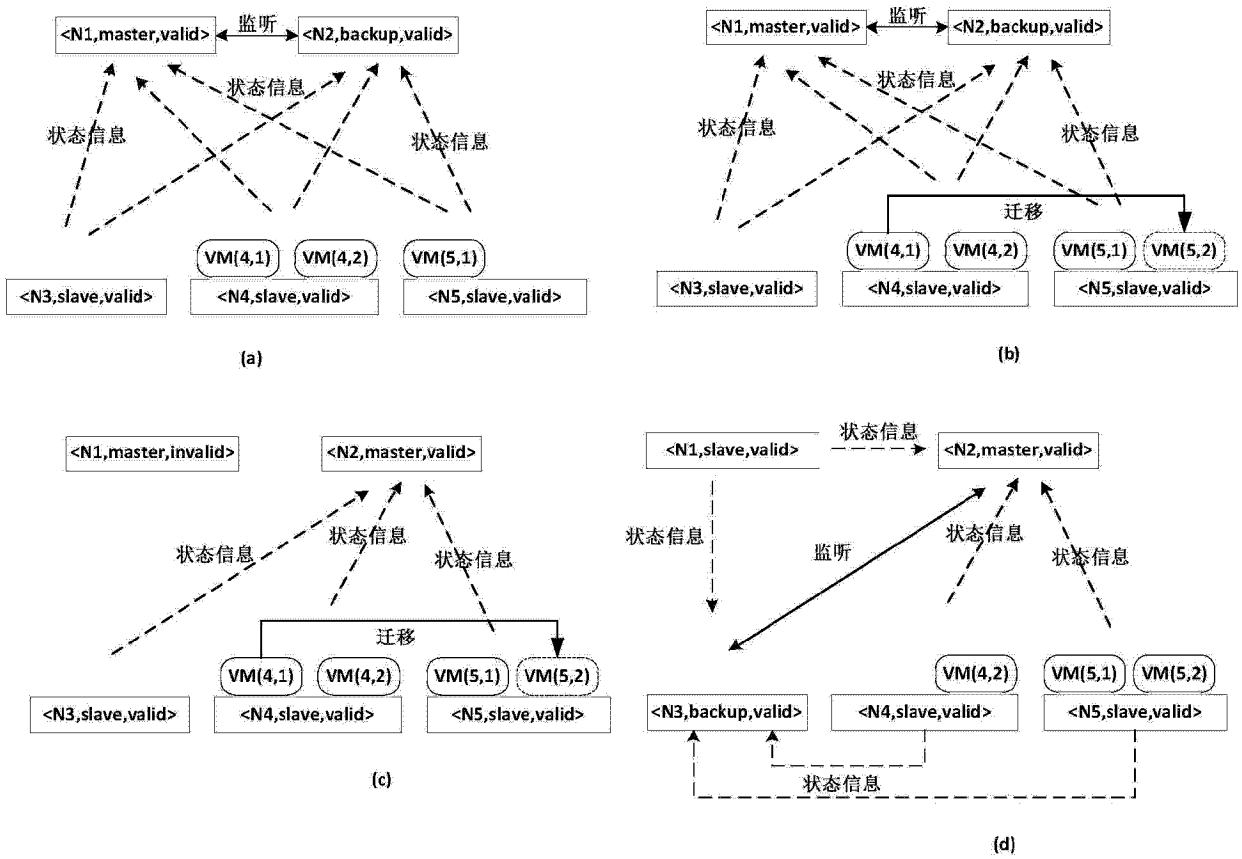


图 10