

分布式锁处理方法、设备、存储介质和系统

本公开要求于 2024 年 01 月 23 日提交中国专利局、申请号为 202410095739.9、申请
5 名称为“分布式锁处理方法、设备、存储介质和系统”的中国专利申请的优先权，其全部
内容通过引用结合在本公开中。

技术领域

本公开涉及云计算技术领域，尤其涉及一种分布式锁处理方法、设备、存储介质
10 和系统。

背景技术

ZooKeeper 是一个开源的分布式协调服务，已经被广泛应用在很多大型分布式系统中，
比如 ZooKeeper 服务系统可以被应用于分布式锁服务的应用场景。ZooKeeper 服务系统中
15 包含多个节点（znode），并可以容忍同时部分节点挂掉，比如 5 个节点构成的系统中可
以容忍同时有 2 个节点挂掉。

一些软件层面的故障可能会导致整个 ZooKeeper 服务系统整体不可用。比如如下的情
形：(1)主节点（Leader 节点）夯住（即运行过程中突然卡住不继续运行后续代码），但
是其与从节点（follower 节点）之间的心跳机制还在正常运行，此时，从节点会认为主节
20 点仍然是正常的，不会触发进行新一轮的主节点选举过程，导致此时服务系统会持续不可
用；(2)假设服务系统中包含 3 个节点，并假设其中的一个从节点所在机器出现硬盘故障
而不可用，另一台主节点 Leader 退出（比如事务 id 耗光、与从节点之间的心跳超时等）
时发生死锁，那么此时 3 个节点中 2 个节点都出现软件故障，会导致服务系统整体不可用。

因此，在使用 ZooKeeper 服务系统的应用场景中，比如分布式锁服务的应
25 用场景，需要保证分布式锁服务的高可用性。

发明内容

本公开实施例提供一种分布式锁处理方法、设备、存储介质和系统，用以提高分布式
锁服务的高可用性。

30 第一方面，本公开实施例提供一种分布式锁处理系统，所述系统包括：
提供分布式锁服务的第一服务集群和第二服务集群，使用所述分布式锁服务的客户端；
所述客户端，用于依次从所述第一服务集群获取第一会话租约，从所述第二服务集群
获取第二会话租约，以所述第一会话租约向所述第一服务集群发送创建锁的第一请求，以
所述第二会话租约向所述第二服务集群发送创建锁的第二请求，以及在收到所述第一服务

集群和所述第二服务集群分别发送的创建锁成功消息时确定获取到锁，所述第一会话租约和所述第二会话租约分别绑定设定的有效期；

所述第一服务集群，用于响应于所述第一请求，在确定没有其他客户端占用所述锁时为所述客户端创建锁，向所述客户端发送创建锁成功消息；

5 所述第二服务集群，用于响应于所述第二请求，在确定没有其他客户端占用所述锁时为所述客户端创建锁，向所述客户端发送创建锁成功消息。

第二方面，本公开实施例提供一种分布式锁处理方法，应用于使用分布式锁服务的客户端，所述方法包括：

10 从第一服务集群获取第一会话租约，以所述第一会话租约向所述第一服务集群发送创建锁的第一请求；

从第二服务集群获取第二会话租约，以所述第二会话租约向所述第二服务集群发送创建锁的第二请求，所述第一服务集群和所述第二服务集群均提供分布式锁服务，所述第一会话租约和所述第二会话租约分别绑定设定的有效期；

15 若接收到所述第一服务集群发送的创建锁成功消息以及接收到所述第二服务集群发送的创建锁成功消息，则确定自己获取到所述锁。

第三方面，本公开实施例提供一种分布式锁处理装置，应用于使用分布式锁服务的客户端，所述装置包括：

第一创建模块，用于从第一服务集群获取第一会话租约，以所述第一会话租约向所述第一服务集群发送创建锁的第一请求；

20 第二创建模块，用于从第二服务集群获取第二会话租约，以所述第二会话租约向所述第二服务集群发送创建锁的第二请求，所述第一服务集群和所述第二服务集群均提供分布式锁服务，所述第一会话租约和所述第二会话租约分别绑定设定的有效期；

确定模块，用于若接收到所述第一服务集群发送的创建锁成功消息以及接收到所述第二服务集群发送的创建锁成功消息，则确定自己获取到所述锁。

25 第四方面，本公开实施例提供一种电子设备，包括：存储器、处理器、通信接口；其中，所述存储器上存储有可执行代码，当所述可执行代码被所述处理器执行时，使所述处理器至少可以实现如第二方面所述的分布式锁处理方法。

30 第五方面，本公开实施例提供了一种非暂时性机器可读存储介质，所述非暂时性机器可读存储介质上存储有可执行代码，当所述可执行代码被电子设备的处理器执行时，使所述处理器至少可以实现如第二方面所述的分布式锁处理方法。

第六方面，本公开实施例提供了一种计算机程序产品，包括计算机程序，所述计算机程序被处理器执行时至少可以实现如第二方面所述的分布式锁处理方法。

35 本公开实施例提供的分布式锁处理系统中，至少包括提供分布式锁服务的第一服务集群和第二服务集群，还包括使用分布式锁服务的客户端。客户群的创建锁（即抢锁）流程如下：客户端依次从第一服务集群获取第一会话租约，从第二服务集群获取

第二会话租约，以第一会话租约向第一服务集群发送创建锁的第一请求，以第二会话租约向第二服务集群发送创建锁的第二请求。第一服务集群响应于第一请求，在确定没有其他客户端占用锁时为该客户端创建锁，向客户端发送创建锁成功消息。同理，第二服务集群响应于第二请求，在确定没有其他客户端占用锁时为客户端创建锁，向客户端发送创建锁成功消息。客户端收到两个创建锁成功的消息，才确定自己抢锁成功。由此可见，每个服务集群维护一把锁，客户端只有依次获得两个服务集群中的锁，才能认定该客户端抢锁成功，也就是说，如果该客户端在某个服务集群中创建锁失败，意味着此时有其他客户端在占用该服务集群提供的锁，则本客户端最终抢锁失败。从而通过提升抢锁阶段的复杂性换取锁更高的健壮性，使得在单个服务集群整体不可用的情况下继续保持分布式锁服务的高可用性。

附图说明

为了更清楚地说明本公开实施例中的技术方案，下面将对实施例描述中所需要使用的附图作一简单地介绍，显而易见地，下面描述中的附图是本公开的一些实施例，对于本领域普通技术人员来讲，在不付出创造性劳动的前提下，还可以根据这些附图获得其他的附图。

- 图1为本公开实施例提供了一种传统分布式锁的抢锁过程的示意图；
- 图2为本公开实施例提供了一种双锁机制的示意图；
- 图3为本公开实施例提供了一种分布式锁处理系统的硬件执行环境的示意图；
- 图4为本公开实施例提供了一种分布式锁处理系统的云计算环境的示意图；
- 图5为本公开实施例提供了一种分布式锁处理方法的执行过程示意图；
- 图6为本公开实施例提供了一种双锁机制下client看到未抢主成功的server的示意图；
- 图7为本公开实施例提供了一种主次锁机制下会话租约的计时设计的示意图；
- 图8为本公开实施例提供了一种心跳延迟响应情形的示意图；
- 图9为本公开实施例提供了一种分布式锁处理方法的流程图；
- 图10为本公开实施例提供了一种分布式锁处理装置的结构示意图；
- 图 11 为本公开实施例提供了一种电子设备的结构示意图。

具体实施方式

为使本公开实施例的目的、技术方案和优点更加清楚，下面将结合本公开实施例中的附图，对本公开实施例中的技术方案进行清楚、完整地描述，显然，所描述的实施例是本公开一部分实施例，而不是全部的实施例。基于本公开中的实施例，本领域普通技术人员在没有作出创造性劳动前提下所获得的所有其他实施例，都属于本公开保护的范围。

需要说明的是，本公开实施例中所涉及的用户信息（包括但不限于用户设备信息、用户个人信息等）和数据（包括但不限于用于分析的数据、存储的数据、展示的数据等），

均为经用户授权或者经过各方充分授权的信息和数据，并且相关数据的收集、使用和处理需要遵守相关地区的相关法律法规和标准，并提供有相应的操作入口，供用户选择授权或者拒绝。

下面结合附图对本公开的一些实施方式作详细说明。在各实施例之间不冲突的情况下，下述的实施例及实施例中的特征可以相互组合。另外，下述各方法实施例中的步骤时序仅为一种举例，而非严格限定。

先对本公开实施例中涉及到的术语或概念进行解释说明：

分布式锁：在分布式系统中用于互斥访问共享资源的机制。它确保在任意时刻只有一个客户端能够获得锁的所有权，从而保证并发操作的正确性。

10 会话租约【**session lease**】：客户端与分布式锁服务之间的会话协议。客户端在抢占锁时创建一个会话租约，会话租约的有效期限决定了锁的有效期。

临时节点：在 **ZooKeeper** 服务集群中用于表示分布式锁的节点。临时节点的生命周期与会话租约绑定，当持有锁的客户端会话超时或停止时，临时节点会被自动删除。

15 分布式锁的可用性：提供分布式锁服务的集群整体不可用时，会话租约会快速过期，导致已经获得锁的客户端进入丢锁逻辑，重新抢占锁。由于该集群不可用，抢锁操作会持续失败，导致抢锁的客户端处于不可用状态。其中，该集群整体不可用，可以是其中的多数节点不可用，比如 **ZooKeeper** 服务集群中包含 3 个 **ZooKeeper** 节点（**znode**），其中至少 2 个节点都不可用时，那么整个
20 **ZooKeeper** 集群不可用。

服务发现：服务发现是指在分布式系统中，自动地发现和识别可用的服务实例，以便其他服务或客户端能够与之进行交互。

目前基于比如 **ZooKeeper** 分布式集群来提供分布式锁服务，已经被广泛使用，此时，常用的 **ZooKeeper** 集群往往采用 3 节点或 5 节点的部署方式，当其中的多数节点不可用时，整个 **ZooKeeper** 集群不可用，从而无法再正常地提供
25 分布式锁服务。本公开实施例提供的分布式锁处理方案，旨在提高分布式锁服务的高可用性。

下面先结合图 1 简单介绍传统的分布式锁的抢锁（即创建锁）的过程以及相关维护过程。

30 分布式锁有两个基本属性：一是所有权，二是有效期。如图 1 所示，以 **ZooKeeper** 分布式锁为例，假设客户端 1 和客户端 2 争抢同一 **ZooKeeper** 集群（图 1 中的 **ZooKeeper** 集群 1）中的同一把锁，每个客户端创建一个会话租约：客户端 1 的会话租约表示为 **lease1**，客户端 2 的会话租约表示为 **lease2**。锁是一种临时节点，它的生命期与会话租约绑定。为便于理解，本实施例中将这个锁的名称命名为 **abc**。
35

假设创建锁的过程如下：1、客户端 1 首先建立与 ZooKeeper 集群 1 的连接（比如 TCP（Transmission Control Protocol，传输控制协议）连接），ZooKeeper 集群 1 为客户端 1 创建会话层的一个标识信息，即为会话租约（lease1），并且绑定对应的有效期，反馈给客户端 1。之后，客户端 1 与 ZooKeeper 集群 1 基于心跳机制进行心跳维护。2、客户端 1 以 lease1 为所有者（owner）标识向 ZooKeeper 集群 1 发送创建锁的请求。假设此时 ZooKeeper 集群 1 发现锁 abc 没有被其他客户端占用，那么 ZooKeeper 集群 1 会创建锁 abc 所对应的临时节点，并关联上客户端 1 的会话租约 lease1，表示客户端 1 此时抢锁成功，且有效期为 lease1 对应的有效期，之后，ZooKeeper 集群 1 向客户端 1 反馈创建锁成功的消息，客户端 1 确定自己抢锁成功。

另外，在客户端 1 执行上述创建锁的过程中，假设客户端 2 也在执行创建锁的过程。如图 1 所示：1、客户端 2 先建立与 ZooKeeper 集群 1 的 TCP 连接，ZooKeeper 集群 1 为客户端 2 创建会话层的一个标识信息，即为会话租约（lease2），并且绑定对应的有效期，反馈给客户端 2。之后，客户端 2 与 ZooKeeper 集群 1 基于心跳机制进行心跳维护。2、客户端 2 以 lease2 为所有者标识向 ZooKeeper 集群 1 发送创建锁的请求，由于此时 ZooKeeper 集群 1 发现锁 abc 正在被客户端 1 占用，那么 ZooKeeper 集群 1 会向客户端 2 反馈创建锁失败的消息，客户端 2 确定自己抢锁失败。

由此可见，同一时间只有一个客户端能抢到锁，从而满足分布式锁的互斥性要求。另外，当持有锁的客户端会话心跳停止后，其会话租约会在预定时间内超时，其占有的锁也会随之被自动释放，其他存活的客户端会顺利抢占，这个确保了锁的存活性。

比如，图 1 所示实施例中，当客户端 1 抢锁成功后的某时间与 ZooKeeper 集群 1 之间的心跳停止（比如不能按照设定的心跳发送间隔向 ZooKeeper 集群发送心跳包），那么以最后一次收到心跳包的响应消息为计时起点，在计时达到 lease1 对应的有效期时，客户端 1 的 lease1 超时，锁 abc 被主动释放，此时 ZooKeeper 集群 1 会将 lease1 及其关联的锁 abc 删除。如图 1 所示：3、假设客户端 2 与 ZooKeeper 集群 1 之间的心跳一直正常，客户端 2 一直在尝试以自己的 lease2 创建锁，所以当此时 ZooKeeper 集群 1 接收到客户端 2 的创建锁的请求时，由于此时并没有其他客户端占用锁 abc，所以将锁 abc 分配给客户端 2 使用：创建锁 abc 的临时节点，并为其关联上客户端 2 的 lease2，有效期即为 lease2 的有效期。

在上述过程中，其实涉及到 lease 的续租问题。简单举例来说，以客户端 1 为例，在其与 ZooKeeper 集群 1 建立 TCP 连接，分配到 lease1 之后，便可以以设定时间间隔向 ZooKeeper 集群 1 发送心跳包，ZooKeeper 集群 1 收到心跳包后

反馈对应的响应消息，从而客户端 1 与 ZooKeeper 集群 1 确定两者之间的通信连接正常。假设客户端 1 收到的 ZooKeeper 集群 1 告知自己的 lease1 的有效期为 20 秒，客户端 1 在收到 ZooKeeper 集群 1 反馈的创建锁成功的消息后开始进行该有效期的计时，假设此时的计时起点为 T1 时刻。在客户端 1 向 ZooKeeper 集群 1 发送下一个心跳包，并接收到对应的响应消息（假设此时为 T2）时，重置计时起点为 T2，此时相当于续约 lease1。假设之后客户端 1 没有再发出心跳包或没有接收到对应的响应消息，则自 T2 开始计时 20 秒，确定 lease1 超时。

在图 1 所示例子中，分布式锁的所有权以及有效期均是通过会话租约来体现的，即两者是耦合在一起。从而产生的可用性问题是：如果提供分布式锁服务的 ZooKeeper 集群 1 整体不可用，那么会话租约很快会过期，抢占到锁的所有权的客户端很快进入丢锁逻辑，进而触发重新抢锁，而因为此时 ZooKeeper 集群 1 不可用，各个客户端的抢锁会持续失败，从而各个客户端也将持续处于不可用状态。

“抢锁”是个低频操作，因此很自然的想法是，能否在这个场景下做到：存量的分布式锁的有效期的维护不受影响，而仅仅影响全新的抢锁操作。这样便可以有效控制提供分布式锁服务的集群的故障爆炸半径。基于此思路，本公开实施例提供的思路是提升抢锁的难度，进而放宽维护锁有效期的门槛，或者说是，增加抢锁的复杂性，换取了锁的健壮性。

基于此，本公开实施例提供了基于主次锁续约机制的高可用的分布式锁处理方案，使得在单个 ZooKeeper 集群整体不可用的情况下继续保持锁服务高可用性。基于 ZooKeeper 集群引入主次锁设计，通过提升抢锁阶段的复杂性换取锁更高的健壮性。该 ZooKeeper 集群仅为一种分布式系统的举例，不以此为限。

如图 2 所示，概括来说，将提供分布式锁服务的服务集群由一个扩展为至少两个，比如图 2 中的 ZooKeeper 集群 1 和 ZooKeeper 集群 2。将每个分布式锁拆成主锁、次锁两把锁，两个锁分别在独立的两个 ZooKeeper 集群中单独维护。实际上，主锁和次锁只是为了区分位于两个集群中的锁，其名称可以是相同的，比如都是锁 abc。此时，客户端要抢到锁的所有权，必须依次获得主、次两把锁的所有权。分布式锁的互斥性体现在同一时刻，至多只有一个客户端能够依次占据两把锁，分布式锁的有效期的有效期则是两把锁各自对应的会话租约的有效期的最大值，即任意一把锁还在有效期内，那么客户端可以认为锁并没有丢，可以继续占锁干活。反之，客户端要争抢分布式锁的前提条件变得更加苛刻了：两把锁的有效期限皆失效。

下面对本公开实施例提供的分布式锁处理方案进行介绍说明。

图 3 为本公开实施例提供的一种分布式锁处理系统的硬件执行环境的示意图，如图 3 所示，该分布式锁处理系统的硬件执行环境可以包括客户端设备 301、

第一服务集群 302 和第二服务集群 303, 客户端设备 301 与第一服务集群 302 和第二服务集群 303 通信连接。

5 第一服务集群 302 和第二服务集群 303 是提供相同分布式锁服务的不同服务集群。每个服务集群中可以包括多个节点, 比如第一服务集群 302 中包括节点 a1、节点 a2 和节点 a3; 第二服务集群 303 中包括节点 b1、节点 b2 和节点 b3。实际应用中, 这两个服务节点可以具体实现为第一 ZooKeeper 集群和第二 ZooKeeper 集群, 每个 ZooKeeper 集群中包括 3 个 ZooKeeper 节点, 即由这 3 个节点对外提供分布式锁服务。本公开实施例中, 可以不区分每个 ZooKeeper 集群中各个节点的主从节点角色。

10 实际应用中, 第一服务集群 302 和第二服务集群 303 可以是基于云端的云服务器构成的两个服务集群, 同一服务集群中的不同节点可以部署在不同的云服务器中。

客户端设备 301, 是指使用第一服务集群 302 和第二服务集群 303 提供的分布式锁服务的客户端, 泛指使用该分布式锁服务的任一客户端。实际应用中, 15 该客户端设备 301 往往是提供某种应用/内容服务的服务器, 比如提供电商、音视频、人工智能等服务的服务器。

实际应用中, 上述第一服务集群 302 和第二服务集群 303 中包含的节点可以是云服务商维护的云服务器—称为计算节点。在如图 4 所示的云计算环境中, 可以包括分布式部署的若干(图 4 中示意的 401-1, 401-2, ...) 计算节点, 每个 20 计算节点中都具有计算、存储等处理资源。在云计算环境中, 可以组织由多个计算节点来提供某种服务; 当然, 一个计算节点也可以提供一种或多种服务, 比如图 4 中示意的服务 A、服务 B、服务 C、服务 D, 其中服务 B 比如是分布式锁服务。也就是说, 在一个计算节点中可以包括除分布式锁服务的其他服务。云计算环境中提供该服务的方式可以是对外提供服务接口, 客户端设备调用该 25 服务接口以使用相应的服务。服务接口包括软件开发工具包 (Software Development Kit, 简称 SDK)、应用程序接口 (Application Programming Interface, 简称 API) 等形式。

在操作过程中, 执行来自客户端设备的请求, 可能需要调用云计算环境中的一个或多个服务, 执行一个服务的一个或多个功能可能需要调用另一个服务的一个或多个功能。如图 4 所示, 服务 A 接收到客户端设备发出的请求后, 可以调用服务 B, 服务 B 可以请求服务 D 执行一个或多个功能。 30

第一服务集群 302 和第二服务集群 303 的每个服务集群中可以包括提供分布式锁服务的多个计算节点, 图 4 中仅示意了一个计算节点 401-1 提供分布式锁服务的情况, 实际上, 可以由多个提供分布式锁服务的计算节点构成一个服务集群, 由提供该分布式锁服务的另外多个计算节点构成另一个服务集群。 35

上述分布式锁服务是根据云计算环境所支持的各种虚拟化技术来部署的，比如基于虚拟机、容器的虚拟化技术。以基于容器的虚拟化技术为例，一个服务对应的若干容器可以被组装成一个容器组（pod）。比如图 4 中示意的服务 B（分布式锁服务）可以配置有一个或多个 pod，每个 pod 内可以包括代理以及一个或多个容器。pod 中的一个或多个容器用于处理与服务的一个或多个相应功能相关的请求，pod 中的代理用于控制与服务相关的网络功能，比如路由、负载均衡等。

基于图 3 所示分布式锁处理系统，客户端设备 301 通过第一服务集群 302 和第二服务集群 303 抢锁即创建锁的过程如下：

10 客户端：依次从第一服务集群获取第一会话租约，从第二服务集群获取第二会话租约，以第一会话租约向第一服务集群发送创建锁的第一请求，以第二会话租约向第二服务集群发送创建锁的第二请求，以及在收到第一服务集群和第二服务集群分别发送的创建锁成功消息时确定获取到锁，第一会话租约和第二会话租约分别绑定设定的有效期；

15 第一服务集群：响应于第一请求，在确定没有其他客户端占用锁时为客户端创建锁，向客户端发送创建锁成功消息；

第二服务集群：响应于第二请求，在确定没有其他客户端占用锁时为客户端创建锁，向客户端发送创建锁成功消息。

丢锁过程如下：客户端若基于第一会话租约和第二会话租约的有效期，确定目标会话租约超时，则确定丢弃锁，其中，目标会话租约是指第一会话租约和第二会话租约中晚超时的会话租约。

其中，第一会话租约和第二会话租约的有效期相同。

结合图5来示例说明上述抢锁和丢锁过程。

抢锁的过程如下：1、客户端1建立与ZooKeeper集群1的TCP连接，ZooKeeper集群1为客户端1创建第一会话租约（lease1），并且绑定对应的有效期，反馈给客户端1。之后，客户端1与ZooKeeper集群1基于心跳机制进行心跳维护。客户端1以lease1为所有者（owner）标识向ZooKeeper集群1发送创建锁的第一请求。假设此时ZooKeeper集群1发现本地的锁abc没有被其他客户端占用，那么ZooKeeper集群1会创建锁abc所对应的临时节点，并关联上客户端1的会话租约lease1，且有效期为lease1对应的有效期，之后，ZooKeeper集群1向客户端1反馈创建锁成功消息。

30 之后，2、在客户端1建立与ZooKeeper集群2的TCP连接，ZooKeeper集群2为客户端1创建第二会话租约（lease2），并且绑定对应的有效期，反馈给客户端1。之后，客户端1与ZooKeeper集群2基于心跳机制进行心跳维护。客户端1以lease2为owner标识向ZooKeeper集群2发送创建锁的第二请求，由于此时ZooKeeper集群2发现本地的锁abc未被其他客户端占用，那么ZooKeeper集群2会向客户端1反馈创建锁成功消息。

35 客户端1由于依次从ZooKeeper集群1和 ZooKeeper集群2都收到了创建锁成功消息，因

此确定自己抢锁成功。

之后，如图5中所示，3、当客户端1抢锁成功后的某时间与ZooKeeper集群1之间的心跳停止/超时，在计时达到lease1的有效期时，ZooKeeper集群1确定客户端1丢锁，删除客户端1对本地的锁abc的占用记录；但是假设此时客户端1与ZooKeeper集群2之间的心跳
5 仍旧正常，也就是说ZooKeeper集群2本地的锁abc仍旧被客户端1占用。此时，客户端1因为lease2还未失效，所以仍旧占用锁abc。与此同时：4、假设客户端2建立与ZooKeeper集群1的TCP连接，ZooKeeper集群1为客户端2创建第三会话租约（lease3），并且绑定对应的有效期，反馈给客户端2。之后，客户端2与ZooKeeper集群1基于心跳机制进行心跳维护。客户端2以lease3为owner标识向ZooKeeper集群1发送创建锁的请求，此时ZooKeeper集群1
10 发现本地的锁abc没有被其他客户端占用，那么ZooKeeper集群1会创建锁abc所对应的临时节点，并关联上客户端2的会话租约lease3，ZooKeeper集群1向客户端2反馈创建锁成功消息。之后，客户端2建立与ZooKeeper集群2的TCP连接，ZooKeeper集群2为客户端2创建第四会话租约（lease4），并且绑定对应的有效期，反馈给客户端2。之后，客户端2与ZooKeeper集群2基于心跳机制进行心跳维护。5、客户端2以lease4为owner标识向ZooKeeper集群2发
15 送创建锁的请求，此时ZooKeeper集群2发现本地的锁abc仍旧被客户端1占用，所以确定客户端2抢锁失败，向客户端2反馈抢锁失败消息。

可以理解的是，在上述举例中，如果客户端1与ZooKeeper集群2之间的心跳也超时，导致lease2对应的有效期计时结束时，客户端1确定丢弃了占用的锁abc。

上述心跳超时/停止，是指客户端与ZooKeeper集群之间基于心跳机制发现通信连接异常的情况。心跳超时，比如是客户端发出心跳包后，在设定时间内未收到ZooKeeper集群
20 反馈的响应消息；心跳停止，比如是客户端隔设定时间后未发出心跳包。

由此可见，在主次锁（比如上述举例中两个ZooKeeper集群本地维护的锁abc一个为主锁，一个为次锁）的双锁机制下判定抢锁成功的条件是同时抢占到两把锁，这等价于同时持有两把锁的有效期。因此，所谓主锁和次锁其实是等价的，交换主、次锁的抢占顺序并不影响锁的互斥性。
25

基于上述主次锁机制，可以实现分布式锁服务的高可用性。因为针对某客户端来说，在主锁或次锁其中一个失效的情况下，仍旧保持另一个锁的会话租约续约（只有心跳正常即一直续约），不丢锁，那么该客户端可以继续工作。而如果只有一套分布式锁服务，那么相应的服务集群不可用时，客户端会丢掉唯一的一个锁（比如图1所示实施例的情况），
30 客户端需要立即重新抢锁，且因为该唯一的一个服务集群不可用，会一直无法抢到锁，客户端就处于停止工作的状态了。

需要说明的是，本公开实施例中，仅以服务集群的粒度对分布式锁的处理过程进行了说明，实际上，以上述ZooKeeper集群1为例，其中可以包括多个节点，客户端与ZooKeeper集群1建立TCP连接实际上是与其中某个节点建立TCP连接，客户端创建锁成功是指在该节点上创建锁成功。只是对使用分布式锁服务的各客户端来说，一个服务集群中各个节点可
35

以设为是等价的，不需要区分。

以上实施例介绍的不需要区分主锁和次锁的获取顺序（也就是可以默认设置一种从两个服务集群抢锁的顺序即可，既可以设置为先从第一服务集群抢锁之后从第二服务集群抢锁，也可以设置为先从第二服务集群抢锁之后从第一服务集群抢锁）的方案，可以适用于使用分布式锁服务的不同客户端具有相等的角色的场景中，比如各个客户端是分布式存储系统中角色相同的各个存储节点。但是，在选主和服务发现应用场景中，是需要区分主锁和次锁的获取顺序的。

其中，选主和服务发现应用场景是指：使用分布式锁服务的多个客户端，通过抢锁实现主从角色的确定——抢锁成功的客户端为主客户端，其他客户端为备客户端，而且，抢锁成功的主客户端将自己的服务地址（比如IP（Internet Protocol，互联网协议）地址）注册到提供分布式锁服务的服务集群中，以方便被访问者发现其需要访问的服务地址。此时，使用分布式锁服务的客户端可以表示为：server（即应用服务器，比如电商服务器、音视频服务器等），而访问应用服务器的可以表示为client（即应用客户端），从而，选主即为多个server中确定主server和备server。

在选主和服务发现场景中，server通过分布式锁完成抢主后，需要再将自身服务地址注册到提供分布式锁服务的服务集群（比如ZooKeeper集群）上以方便被client访问。因此，在一可选实施例中，server可以在抢占锁的同时将自身服务地址写入服务集群，这样就同时完成了抢锁和服务地址注册的逻辑。在此应用场景下，使用上述双锁机制，若通过主锁进行服务地址注册，部分场景下会存在client看到了抢主失败的备server注册的服务地址，但该备server实际还没有开始工作的情形。

下面结合图6示例说明上述情形，在图6中，假设server1通过图中示意的步骤1和步骤2先后在ZooKeeper集群1和ZooKeeper集群2抢锁成功（具体过程参考图5所示实施例中相关说明，在此不赘述），并假设从ZooKeeper集群1中抢到的锁称为主锁，从ZooKeeper集群2中抢到的锁称为次锁，实际上，主、次锁这是存储在两个服务集群中的同一把锁：比如上述举例中的锁abc。而且，server1抢到主锁时将自己的服务地址注册到ZooKeeper集群1。从而如图6中所示，client通过ZooKeeper集群1可以查询到server1的服务地址。

之后，server1抢到主、次锁后开始工作，一段时间后，如图6中所示，假设server1与ZooKeeper集群1出现心跳超时现象，即主锁维持心跳超时，server2此时从ZooKeeper集群1中抢到主锁，并注册自己的服务地址到ZooKeeper集群1。此时，因为server1依然维持次锁的会话租约，即次锁还在有效期，所以server2从ZooKeeper集群2中抢占次锁会失败（可以参考图5所示实施例中相关描述，在此不赘述）。也就是说，此时仍旧是server1在占用锁，那么client在ZooKeeper集群1中看到的应该是目前在工作中的server1的服务地址，但是因为server2在ZooKeeper集群1抢主锁成功，将其服务地址注册到ZooKeeper集群1中，使得ZooKeeper集群1认为server2此时为主server，将此前注册的server1的服务地址替换为server2的服务地址，此时client从ZooKeeper集群1看到的是server2的服务地址，这样

便产生了选主不一致的问题：应该选出的主server为server1，但是实际上注册服务地址的是备server2。

为了解决在选主和服务发现场景下存在的上述选主不一致问题，相比于前述实施例
5 第一服务集群和第二服务集群不区分主、次锁的情形：默认设置第一服务集群为主服务集群，提供的是主锁，将第二服务集群设置为次服务集群，提供的是次锁，默认server先抢占主锁再抢占次锁，抢占主锁成功后将服务地址注册到主服务集群即第一服务集群。本公开实施例提供了如下解决选主不一致的方式：将交换主、次锁的默认抢占顺序，在一server持有了次锁的所有权后再去抢占主锁。

具体地，将第二服务集群被配置为提供服务地址注册服务的服务集群（与前述实施例
10 不同，这里将其提供的锁称为主锁），将第一服务集群配置为提供次锁的服务集群。实际上，由于主次锁仅是命名上区分位于不同服务集群中的同一把锁，也可以统称为第一锁、第二锁，从而该解决方式本质是：server先抢未提供服务地址注册服务的第一服务集群所提供的锁（次锁），之后再抢提供服务地址注册服务的第二服务集群所提供的锁（主锁），并在抢主锁成功后将server的服务地址注册到第二服务集群中。

15 针对各server，默认配置了提供服务地址注册功能的服务集群为第二服务集群。所以，使用分布式锁服务的客户端（即server），在接收到第二服务集群发送的创建锁成功消息时，将服务地址注册到第二服务集群，以使该server的访问对象（client）从第二服务集群获取该server的服务地址。

在上述解决选主一致性问题的方式下，仍以图6中示意的server1和server2抢锁的情
20 形为例来说，server1先从ZooKeeper集群1中抢锁成功，之后再从ZooKeeper集群2中抢锁，若从ZooKeeper集群2中也抢锁成功，则将自己的服务地址注册到ZooKeeper集群2。此时，client通过ZooKeeper集群2看到的是server1的服务地址。之后，假设server1在ZooKeeper集群1中抢到的锁被释放（比如心跳停止或心跳超时导致的），但是仍旧持有从ZooKeeper集群2中抢到的锁，而此时server2从ZooKeeper集群1中抢锁成功，但是在尝试从ZooKeeper
25 集群2抢锁时，因为server1占用着锁，所以server2抢主锁失败，从而保持client从ZooKeeper集群2中看到的仍旧是持有锁的server1的服务地址。

值得说明的是，如果先抢占次锁，为了能保证抢夺主锁成功时，次锁的会话租约依然有效，可在尝试抢主锁前，保证次锁的剩余有效期大于抢占主锁所需要花费的时间，以便等待抢占主锁成功的消息返回。极端情况下，server如果接收到第二服务集群反馈的抢占
30 主锁成功的消息后发现次锁的会话租约到期，server需要主动释放主锁，进入抢占次锁的逻辑。这样便可以保证client解析到的是真正的主server的服务地址。

以上实施例提供的是分布式锁处理系统中仅包括提供分布式锁服务的第一服务集群
和第三服务集群这两个服务集群的情况。实际上，为了进一步提高分布式锁服务的高可用性，还可以将提供分布式锁服务的服务集群从2个扩展为更多个，比如3个、5个。

35 也就是说，分布式锁处理系统除第一服务集群和第二服务集群外，还包括提供分布式

锁服务的至少一个其他服务集群。此时，使用分布式锁服务的客户端，在接收到至少目标数量的服务集群发送的创建锁成功消息时，才确定自己获取到锁。其中，该目标数量大于服务集群数量的一半。

此时，扩展后的分布式锁的安全性和活性主要体现为：

5 分布式锁的安全性：当同一个客户端抢占了多数派的锁资源后，则认为抢锁成功，如果多数派续约超时则认为丢锁。其中，多数派即为整体服务集群中的过半的服务集群。比如一共3个服务集群，多数派即为其中的至少2个服务集群，比如一共5个服务集群，多数派即为其中的至少3个服务集群。

10 分布式锁的活性：（1）不出现死锁，服务集群的锁资源会自动过期被回收；每个客户端依照固定次序抢占各服务集群的锁资源，抢占失败则跳过抢占下一个服务集群的锁资源，每次抢占锁失败都需要退避比成功抢占多数派锁耗时更长的时间，方便已经抢占锁资源成功的客户端优先争抢后续锁资源。（2）单个服务集群宕机不影响整体分布式锁的可用性。

15 前述各实施例中提到会话租约一方面作为会话层上使用分布式锁服务的客户端的一种标识，另一方面，其还绑定有效期，在有效期超时时，所占用的锁会被释放。这就涉及到对会话租约的有效期的计时机制的设计。

20 在传统的分布式锁的方案中，比如redis的redlock算法中，采用的计时方式都是基于物理时钟的这种绝对时间的计时方式，比如采用网络时间协议（Network Time Protocol，简称NTP）时间。物理时钟可能发生漂移，即提供分布式锁服务的服务集群与使用分布式锁服务的客户端之间的物理时钟可能发生漂移，该时钟漂移的发生会破坏锁的互斥性。

25 首先需要说明一点，对于服务集群和客户端来说，都是需要进行同一会话租约的有效期的计时的，比如某客户端对应的会话租约，有效期为20秒，则客户端和服务集群都需要进行这个会话租约的有效期的计时。从而，比如，当服务集群的物理时钟发生向前漂移（未来某个时刻，即时间走的过快），使得服务集群端认为本端会话租约的有效期提前达到（即锁提前超时），但是此时客户端端会话租约的有效期还未到达，即认为自己还持有锁。此时，另一客户端再次向该服务集群抢锁，由于服务集群已经认为前一客户端的会话租约到期，所以会将锁分配给该另一客户端，这样就导致锁同时被两个客户端持有，打破了锁的互斥性。再比如，客户端的物理时钟发生了向后漂移（过去某个时刻，即时间走慢了），此时会出现客户端认为自己还持有锁（本端会话租约有效期未到），但是服务集群已经判
30 断该客户端丢锁（本端该会话租约有效期已到）的情况，同样会出现上述多个客户端同时持有锁的情况发生，打破锁的互斥性。上述举例的两种情形都是客户端端会话租约晚于服务集群端会话租约过期的事情的发生，导致锁互斥性被破坏。

35 为了克服上述物理时钟的计时方式对锁的互斥性性能的影响，本公开实施例提供的主次锁设计上，会话租约的有效期的计时机制不依赖物理时钟，而是依靠的是（在短时间内的）时钟频率一致的现象，换言之，任何两台机器，相

同时间里流逝时长可以认为是一样的。正是基于这样一个判断，在上述主次锁机制下，客户端和各服务集群端的会话租约的时间都是单调的相对时间，没使用绝对时间，这样可以保证会话租约维护的准确性，保证客户端会话租约早于服务集群端会话租约而过期。因为如果某客户端认为本端维护的会话租约已经到期，但是服务集群端该会话租约还未到期，那么客户端释放占用的锁，服务集群认为该客户端仍旧占用锁而拒绝其他客户端的创建锁请求（即抢锁请求），同一时刻锁仅被一个客户端占用，不会破坏锁的互斥性。

在具体实施时，服务集群和客户端都可以采用 CPU（Central Processing Unit，中央处理器）滴答周期（CPU tick）作为相对时间的计时方式来对会话租约的有效期进行计时。实际上，可以根据 CPU tick 确定 MonotonicTime（单调时间），而会话租约的有效期的计时通过计数 MonotonicTime 来实现。根据 CPU tick 确定 MonotonicTime 的方式可以参考现有相关技术实现。

承接前述实施例，针对客户端来说，客户端以接收到第一服务集群发送的创建锁成功消息的时刻作为第一计时起点，根据 CPU 滴答周期对第一会话租约的有效期进行计时，以及，以接收到第二服务集群发送的创建锁成功消息的时刻作为第二计时起点，根据 CPU 滴答周期对第二会话租约的有效期进行计时。

针对第一服务集群和第二服务集群来说，第一服务集群以接收到客户端发送的起始心跳包的时刻重置本地维护的第一会话租约的有效期的第三计时起点，根据 CPU 滴答周期对本地维护的第一会话租约的有效期进行计时，在确定本地维护的第一会话租约超时时删除锁和第一会话租约。类似地，第二服务集群以接收到客户端发送的起始心跳包的时刻重置本地维护的第二会话租约的有效期的第四计时起点，根据 CPU 滴答周期对本地维护的第二会话租约的有效期进行计时，在确定本地维护的第二会话租约超时时删除锁和第二会话租约。可以理解的是，客户端在接收到第一服务集群反馈的创建锁成功消息后，会发送起始心跳包（是指最初向第一服务集群发送的心跳包）至第一服务集群，第一服务集群收到该起始心跳包时重新进行本端的第一会话租约的有效期的计时。同理，客户端在接收到第二服务集群反馈的创建锁成功消息后，会发送起始心跳包（是指最初向第二服务集群发送的心跳包）至第二服务集群，第二服务集群收到该起始心跳包时重新进行本端的第二会话租约的有效期的计时。

其中，第一服务集群和第二服务集群对各自对应的第一会话租约、第二会话租约的有效期的起始计时时刻分别是：第一服务集群创建出第一会话租约的时刻，第二服务集群创建出第二会话租约的时刻。

另外，客户端只有在本端计时确定第一会话租约和第二会话租约都过期时，才确定自己释放锁，进而进入下一次的抢锁逻辑。

另外，需要说明的是，客户端自第一计时起点，以设定时间间隔向第一服务集群发送心跳包，若接收到第一服务集群反馈的响应消息则重置第一计时起点为接收到该响应消息

的时刻；以及，自第二计时起点，以设定时间间隔向第二服务集群发送心跳包，若接收到第二服务集群反馈的响应消息则重置第二计时起点为接收到该响应消息的时刻。可以理解的是，第一和第二计时起点重置后，相当于从新的计时起点再重新计时相应会话租约的有效期，此时实际上是实现了会话租约的一次续约。当然，下一心跳包和响应消息的正常收发也会再次导致重置计时起点。

同理，第一服务集群和第二服务集群进行本端的第一会话租约和第二会话租约的有效期的计时过程中，每当收到客户端发送的心跳包，反馈对应的响应消息时，也会重置本地的上述第三计时起点、第四计时起点。

下面结合图7来说明会话租约的有效期的计时机制。如图7所示，在客户端侧的会话租约有效期与服务集群端的相同会话租约的有效期的时长一致情况下，在前述主次锁机制下，客户端是从收到ZooKeeper集群1反馈的创建锁成功消息时（此时作为第一计时起点）开始对第一会话租约（图中表示为lease1）的有效期进行计时的，从收到ZooKeeper集群2反馈的创建锁成功消息时（此时作为第二计时起点）开始对第二会话租约（图中表示为lease2）的有效期进行计时的。ZooKeeper集群1和ZooKeeper集群2均在收到客户端分别发送的起始心跳包后开始进行相应的lease1和lease2的有效期的超时计时（分别对应图中的第三计时起点、第四计时起点），因此客户端和ZooKeeper集群1、ZooKeeper集群2的计时起点由于因果性（客户端收到创建锁成功消息后发送起始心跳包）保证了先后顺序，再通过单调的相对时间计时（即CPU tick的计时方式），可以保证在客户端不再发起续约（即不能进行心跳包的正常收发）时，客户端的会话租约总是先于服务集群端的会话租约的有效期而超时。比如图7中示意的：假设客户端自分别向ZooKeeper集群1、ZooKeeper集群2发生起始心跳包后再没有发送下一个心跳包，那么自第一计时起点计时lease1的有效期时长达到T1时刻时，客户端确定lease1有效期超时，自第二计时起点计时lease2的有效期时长达到T1'时刻时，客户端确定lease2有效期超时。ZooKeeper集群1因为没有再接收到心跳包，自第三计时起点计时lease1的有效期时长达到T2时刻时，确定lease1有效期超时，同理，ZooKeeper集群2因为没有再接收到心跳包，自第四计时起点计时lease2的有效期时长达到T3时刻时，确定lease2有效期超时。而且，T1是早于T2的，T1'是早于T3的，从而基于相对时间的计时方式，实现了客户端会话租约的有效期早于相应服务集群端会话租约的有效期而超时的目的。

以上各实施例都是假设某一会话租约在客户端和各服务集群端的有效期是相等的情况。但是，在实际应用中，客户端与第一服务集群、第二服务集群之间的网络延迟可能有时候会较大，极端情况下，比如客户端发送至第一服务集群的心跳包，第一服务集群可能延迟了第一会话租约的有效期的时长才反馈响应消息，此时会出现客户端认为自己仍旧持有锁（因为在本端第一会话租约的有效期内收到了该响应消息，续约成功），但是第一服务集群确定客户端丢锁（因为第一服务集群已经计时到本端第一会话租约的有效期超时）的情况发生。如图8中所示，客户端在T1时刻向ZooKeeper集群1发送一个心跳包，假设此

时网络延迟很低,可以忽略不计,即假设ZooKeeper集群1在T1时刻收到了该心跳包。之后,假设客户端在延迟了lease1的一倍有效期时长后的T2时刻收到ZooKeeper集群1发送的响应消息,客户端此时确定lease1续约成功,假设随即向ZooKeeper集群1发送下一个心跳包。此时,如图8中所示,ZooKeeper集群1在T1时刻开始计时,计时lease1的一倍有效期时长到T2时刻时,确定客户端的lease1失效,此时便会断开与客户端的通信连接,从而接收不到上述“下一个心跳包”,导致客户端之后的lease1续约失败,丢锁。

为避免这种情况的发生,第一服务集群可以确定本地维护的第一会话租约的有效期是客户端维护的第一会话租约的有效期的至少两倍。同理,第二服务集群确定本地维护的第二会话租约的有效期是客户端维护的第二会话租约的有效期的至少两倍。

也就是说,针对客户端对应的第一会话租约(比如lease1),第一服务集群告知客户端该第一会话租约的有效期比如为20秒,但是第一服务集群本端记录该第一会话租约的有效期为40秒(2倍的情形);同理,针对客户端对应的第二会话租约(比如lease2),第二服务集群告知客户端该第二会话租约的有效期比如为20秒,但是第二服务集群本端记录该第二会话租约的有效期为40秒(2倍的情形)。这样,在上述举例中,即使第一服务集群在本端计时到第一会话租约的40秒有效期已经过半(达到20秒)时才向客户端发送的心跳包的响应消息,第一服务集群也因为该40秒有效期还未超时而确定客户端仍旧持有锁,与客户端的第一会话租约成功续约的情况相匹配。

综上,主次锁设计下,会话租约的有效期的计时不依赖物理时钟,而依赖可信的相对时间,不会受物理时钟漂移影响导致产生客户端的会话租约晚于服务集群端会话租约而过期的事情发生。而且,通过设置服务集群端会话租约的有效期至少2倍于客户端会话租约的有效期,可以降低网络延迟情况对客户端会话租约续约的影响。

图9为本公开实施例提供的一种分布式锁处理方法的流程图,该方法可以由前述实施例中使用分布式锁服务的客户端来执行。如图9所示,该方法包括如下步骤:

901、客户端从第一服务集群获取第一会话租约,以第一会话租约向第一服务集群发送创建锁的第一请求。

902、客户端从第二服务集群获取第二会话租约,以第二会话租约向第二服务集群发送创建锁的第二请求,第一服务集群和第二服务集群均提供分布式锁服务,第一会话租约和第二会话租约分别绑定设定的有效期。

903、客户端若接收到第一服务集群发送的创建锁成功消息以及接收到第二服务集群发送的创建锁成功消息,则确定自己获取到锁。

在一可选实施例中,所述方法还包括:客户端若基于第一会话租约和第二会话租约的有效期,确定目标会话租约超时,则确定丢弃锁,其中,目标会话租约是指第一会话租约和第二会话租约中晚超时的会话租约。

在一可选实施例中,关于会话租约的有效期的计时,客户端以接收到第一服务集群发送的创建锁成功消息的时刻作为第一计时起点,根据CPU滴答周期对第一会话租约的有效

期进行计时，以及，以接收到第二服务集群发送的创建锁成功消息的时刻作为第二计时起点，根据CPU滴答周期对第二会话租约的有效期进行计时。

在一可选实施例中，在选主和服务发现场景下，所述第二服务集群被配置为提供服务地址注册服务，从而，在客户端先从第一服务集群创建锁（即抢锁），后从第二服务集群创建锁的过程中，客户端在接收到第二服务集群发送的创建锁成功消息时，将服务地址注册到第二服务集群，以使客户端的访问对象从第二服务集群获取客户端的服务地址。

以下将详细描述本公开的一个或多个实施例的分布式锁处理装置。本领域技术人员可以理解，这些装置均可使用市售的硬件组件通过本方案所教导的步骤进行配置来构成。

图10为本公开实施例提供的一种分布式锁处理装置的结构示意图，该装置应用于使用分布式锁服务的客户端。如图10所示，该装置包括：第一创建模块11、第二创建模块12、确定模块13。

第一创建模块11，用于从第一服务集群获取第一会话租约，以所述第一会话租约向所述第一服务集群发送创建锁的第一请求。

第二创建模块12，用于从第二服务集群获取第二会话租约，以所述第二会话租约向所述第二服务集群发送创建锁的第二请求，所述第一服务集群和所述第二服务集群均提供分布式锁服务，所述第一会话租约和所述第二会话租约分别绑定设定的有效期。

确定模块13，用于若接收到所述第一服务集群发送的创建锁成功消息以及接收到所述第二服务集群发送的创建锁成功消息，则确定自己获取到所述锁。

图10所示装置可以执行前述实施例中客户端执行的步骤，详细的执行过程和技术效果参见前述实施例中的描述，在此不再赘述。

在一个可能的设计中，上述图10所示分布式锁处理装置的结构可实现为一电子设备。如图11所示，该电子设备可以包括：处理器21、存储器22、通信接口23。其中，存储器22上存储有可执行代码，当所述可执行代码被处理器21执行时，使处理器21至少可以实现如前述实施例中提供的分布式锁处理方法。

另外，本公开实施例提供了一种非暂时性机器可读存储介质，所述非暂时性机器可读存储介质上存储有可执行代码，当所述可执行代码被电子设备的处理器执行时，使所述处理器至少可以实现如前述实施例中提供的分布式锁处理方法。

本公开实施例提供了一种计算机程序产品，包括计算机程序，所述计算机程序被处理器执行时至少可以实现如前述实施例中提供的分布式锁处理方法。

以上所描述的装置实施例仅仅是示意性的，其中所述作为分离部件说明的网元可以是或者也可以不是物理上分开的。可以根据实际的需要选择其中的部分或者全部模块来实现本实施例方案的目的。本领域普通技术人员在不付出创造性的劳动的情况下，即可以理解并实施。

通过以上的实施方式的描述，本领域的技术人员可以清楚地了解到各实施方式可借助加必需的通用硬件平台的方式来实现，当然也可以通过硬件和软件结合的方式来实现。基

于这样的理解，上述技术方案本质上或者说对现有技术做出贡献的部分可以以计算机产品的形式体现出来，本公开可采用在一个或多个其中包含有计算机可用程序代码的计算机可用存储介质（包括但不限于磁盘存储器、CD-ROM（Compact Disc Read-Only Memory，只读光盘）、光学存储器等）上实施的计算机程序产品的形式。

- 5 最后应说明的是：以上实施例仅用以说明本公开的技术方案，而非对其限制；尽管参照前述实施例对本公开进行了详细的说明，本领域的普通技术人员应当理解：其依然可以对前述各实施例所记载的技术方案进行修改，或者对其中部分技术特征进行等同替换；而这些修改或者替换，并不使相应技术方案的本质脱离本公开各实施例技术方案的精神和范围。

权 利 要 求 书

- 1、一种分布式锁处理系统，其中，包括：
- 提供分布式锁服务的第一服务集群和第二服务集群，使用所述分布式锁服务的客户端；
- 所述客户端，用于依次从所述第一服务集群获取第一会话租约，从所述第二服务集群
- 5 获取第二会话租约，以所述第一会话租约向所述第一服务集群发送创建锁的第一请求，以
- 所述第二会话租约向所述第二服务集群发送创建锁的第二请求，以及在收到所述第一服务
- 集群和所述第二服务集群分别发送的创建锁成功消息时确定获取到锁，所述第一会话租约
- 和所述第二会话租约分别绑定设定的有效期；
- 所述第一服务集群，用于响应于所述第一请求，在确定没有其他客户端占用所述锁时
- 10 为所述客户端创建锁，向所述客户端发送创建锁成功消息；
- 所述第二服务集群，用于响应于所述第二请求，在确定没有其他客户端占用所述锁时
- 为所述客户端创建锁，向所述客户端发送创建锁成功消息。
- 2、根据权利要求1所述的系统，其中，所述客户端，用于若基于所述第一会话租约和
- 所述第二会话租约的有效期，确定目标会话租约超时，则确定丢弃所述锁，其中，所述目
- 15 标会话租约是指所述第一会话租约和所述第二会话租约中晚超时的会话租约。
- 3、根据权利要求1或2所述的系统，其中，所述客户端，用于以接收到所述第一服务
- 集群发送的创建锁成功消息的时刻作为第一计时起点，根据中央处理器CPU滴答周期对所
- 述第一会话租约的有效期进行计时，以及，以接收到所述第二服务集群发送的创建锁成功
- 消息的时刻作为第二计时起点，根据CPU滴答周期对所述第二会话租约的有效期进行计时。
- 20 4、根据权利要求3所述的系统，其中，所述客户端，用于自所述第一计时起点，以设
- 定时间间隔向所述第一服务集群发送心跳包，若接收到所述第一服务集群反馈的响应消息
- 则重置所述第一计时起点为接收到所述响应消息的时刻；以及，自所述第二计时起点，以
- 设定时间间隔向所述第二服务集群发送心跳包，若接收到所述第二服务集群反馈的响应消
- 息则重置所述第二计时起点为接收到所述响应消息的时刻。
- 25 5、根据权利要求1-4中任一项所述的系统，其中，所述第一服务集群，用于确定本地
- 维护的所述第一会话租约的有效期是所述客户端维护的所述第一会话租约的有效期的至
- 少两倍；
- 所述第二服务集群，用于确定本地维护的所述第二会话租约的有效期是所述客户端维
- 护的所述第二会话租约的有效期的至少两倍。
- 30 6、根据权利要求5所述的系统，其中，所述第一服务集群，用于以接收到所述客户端
- 发送的起始心跳包的时刻重置本地维护的所述第一会话租约的有效期的第三计时起点，根
- 据CPU滴答周期对本地维护的所述第一会话租约的有效期进行计时，在确定本地维护的所
- 述第一会话租约超时时删除所述锁和所述第一会话租约；
- 所述第二服务集群，用于以接收到所述客户端发送的起始心跳包的时刻重置本地维护
- 35 的所述第二会话租约的有效期的第四计时起点，根据CPU滴答周期对本地维护的所述第二

会话租约的有效期进行计时，在确定本地维护的所述第二会话租约超时时删除所述锁和所述第二会话租约。

7、根据权利要求1-6中任一项所述的系统，其中，所述第二服务集群被配置为提供服务地址注册服务；

5 所述客户端，用于在接收到所述第二服务集群发送的创建锁成功消息时，将服务地址注册到所述第二服务集群，以使所述客户端的访问对象从所述第二服务集群获取所述客户端的服务地址。

8、根据权利要求1-6中任一项所述的系统，其中，所述系统还包括提供所述分布式锁服务的至少一个其他服务集群；

10 所述客户端，用于在向各服务集群发送创建锁的请求后，若接收到至少目标数量的服务集群发送的创建锁成功消息时，确定自己获取到所述锁，所述目标数量大于所述服务集群数量的一半。

9、一种分布式锁处理方法，其中，应用于使用分布式锁服务的客户端，包括：

15 从第一服务集群获取第一会话租约，以所述第一会话租约向所述第一服务集群发送创建锁的第一请求；

从第二服务集群获取第二会话租约，以所述第二会话租约向所述第二服务集群发送创建锁的第二请求，所述第一服务集群和所述第二服务集群均提供分布式锁服务，所述第一会话租约和所述第二会话租约分别绑定设定的有效期；

20 若接收到所述第一服务集群发送的创建锁成功消息以及接收到所述第二服务集群发送的创建锁成功消息，则确定自己获取到所述锁。

10、根据权利要求9所述的方法，其中，所述方法还包括：

若基于所述第一会话租约和所述第二会话租约的有效期，确定目标会话租约超时，则确定丢弃所述锁，其中，所述目标会话租约是指所述第一会话租约和所述第二会话租约中晚超时的会话租约。

25 11、根据权利要求10所述的方法，其中，所述方法还包括：

以接收到所述第一服务集群发送的创建锁成功消息的时刻作为第一计时起点，根据中央处理器CPU滴答周期对所述第一会话租约的有效期进行计时；

以接收到所述第二服务集群发送的创建锁成功消息的时刻作为第二计时起点，根据CPU滴答周期对所述第二会话租约的有效期进行计时。

30 12、根据权利要求9-11中任一项所述的方法，其中，所述第二服务集群被配置为提供服务地址注册服务；

所述方法还包括：

在接收到所述第二服务集群发送的创建锁成功消息时，将服务地址注册到所述第二服务集群，以使所述客户端的访问对象从所述第二服务集群获取所述客户端的服务地址。

35 13、一种电子设备，其中，包括：存储器、处理器、通信接口；其中，所述存储器上

存储有可执行代码，当所述可执行代码被所述处理器执行时，使所述处理器执行如权利要求9至12中任一项所述的分布式锁处理方法。

- 14、一种非暂时性机器可读存储介质，其中，所述非暂时性机器可读存储介质上存储有可执行代码，当所述可执行代码被电子设备的处理器执行时，使所述处理器执行如权利要求9至12中任一项所述的分布式锁处理方法。

15、一种计算机程序产品，包括计算机程序，所述计算机程序被处理器执行时实现如权利要求9至12中任一项所述的方法。