32 | 答疑篇：如何判断并解决网络分区问题？

2019-12-11 聂鹏程

讲述：聂鹏程
时长 14:41 大小 13.46M

你好，我是聂鹏程。今天，我来继续带你打卡分布式核心技术。

到目前为止，“分布式技术原理与算法解析”专栏已经接近尾声了。在这里，我首先要感谢你坚持学习每一篇文章，以及对每一道思考题的积极思考与讨论，并在此基础上扩展了类似问题。

比如 @Jackey、@Eternal、@leslie、@mt11912、@小白啊、@随心而至等同学，一直在跟着专栏的更新节奏学习，并非常积极地在留言区留言讨论、总结自己的理解，并查询相关资料补充文中未讲解到或没有深入展开的问题。

今天，我梳理了文后的留言，发现大家对最近几篇文章介绍的分布式高可靠问题特别感兴趣，特别是我没有详细展开的网络分区问题。

比如，在第 4 篇文章“分布式选举：国不可一日无君”中，我给你留下的思考题是集群中是否会存在双主的场景，很多同学提到双主是网络分区导致的。

再比如，在第 31 篇文章“分布式高可用之故障恢复：知错能改，善莫大焉”中，我给你留下的思考题是，如何判断以及处理网络分区。

因此，在今天这篇文章中，我将会与你深入探讨网络分区问题，以帮助你进一步理解并解决业务中的故障恢复问题。

什么是网络分区？

我们先来看看网络分区到底是什么吧。在第 31 篇文章分享故障恢复时，我与你介绍了故障类型中的网络故障，网络分区就是其中的一种故障类型。

通常情况下，网络分区指的是在分布式集群中，节点之间由于网络不通，导致集群中节点形成不同的子集，子集中节点间的网络相通，而子集和子集间网络不通。也可以说，网络分区是子集与子集之间在网络上相互隔离了。

那么，应该如何判断是否发生了网络分区呢？

如何判断是否发生了网络分区？

在分布式集群中，不同的集群架构网络分区的形态略有不同。所以，要判断是否发生了网络分区，我们需要弄清楚不同的分布式集群架构，即集中式架构和非集中式架构中的网络分区形态是什么样的。

首先，我们来看一下**集中式架构的网络分区形态**。

集中式架构中，Master 节点通常以一主多备的形式部署，Slave 节点与 Master 节点相连接，Master 节点的主和备之间会通过心跳相互通信。

以 Master 节点主备部署为例，如下图所示，集中式架构中的网络分区主要是主节点与备节点之间网络不通，且一部分 Slave 节点只能与主 Master 节点连通，另一部分只能与备 Master 节点连通。

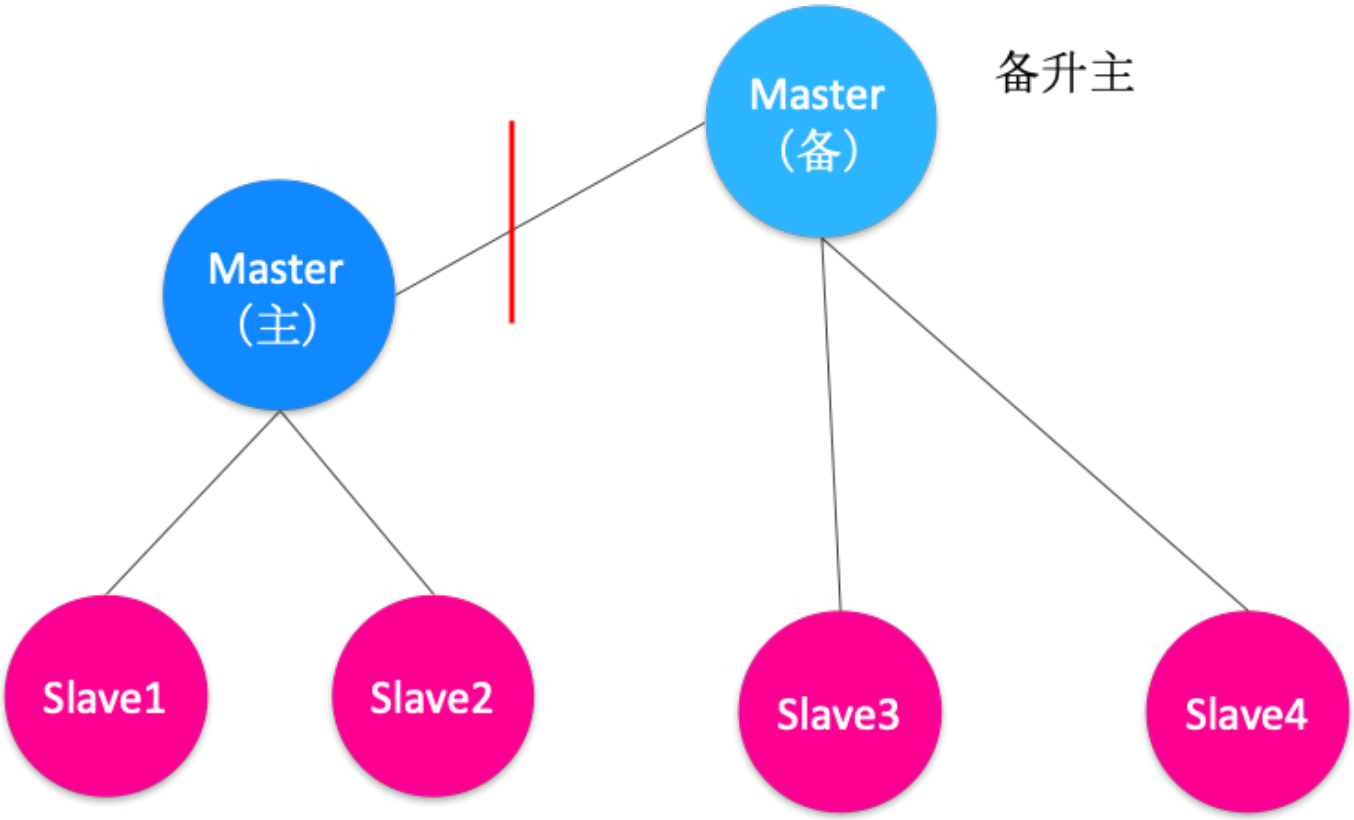

图 1 集中式架构中网络分区的形态

然后，我们再来看看**非集中式架构中的网络分区形态**。

如下图所示，非集中式架构中，节点是对称的，因此网络分区的形态是形成不同子集，子集内节点间可互相通信，而子集之间的节点不可通信。比如，子集群 1 中 Node1、Node2 和 Node4 可以相互通信，子集群 2 中 Node3 和 Node5 也可以相互通信，但子集群 1 和子集群 2 之间网络不通。

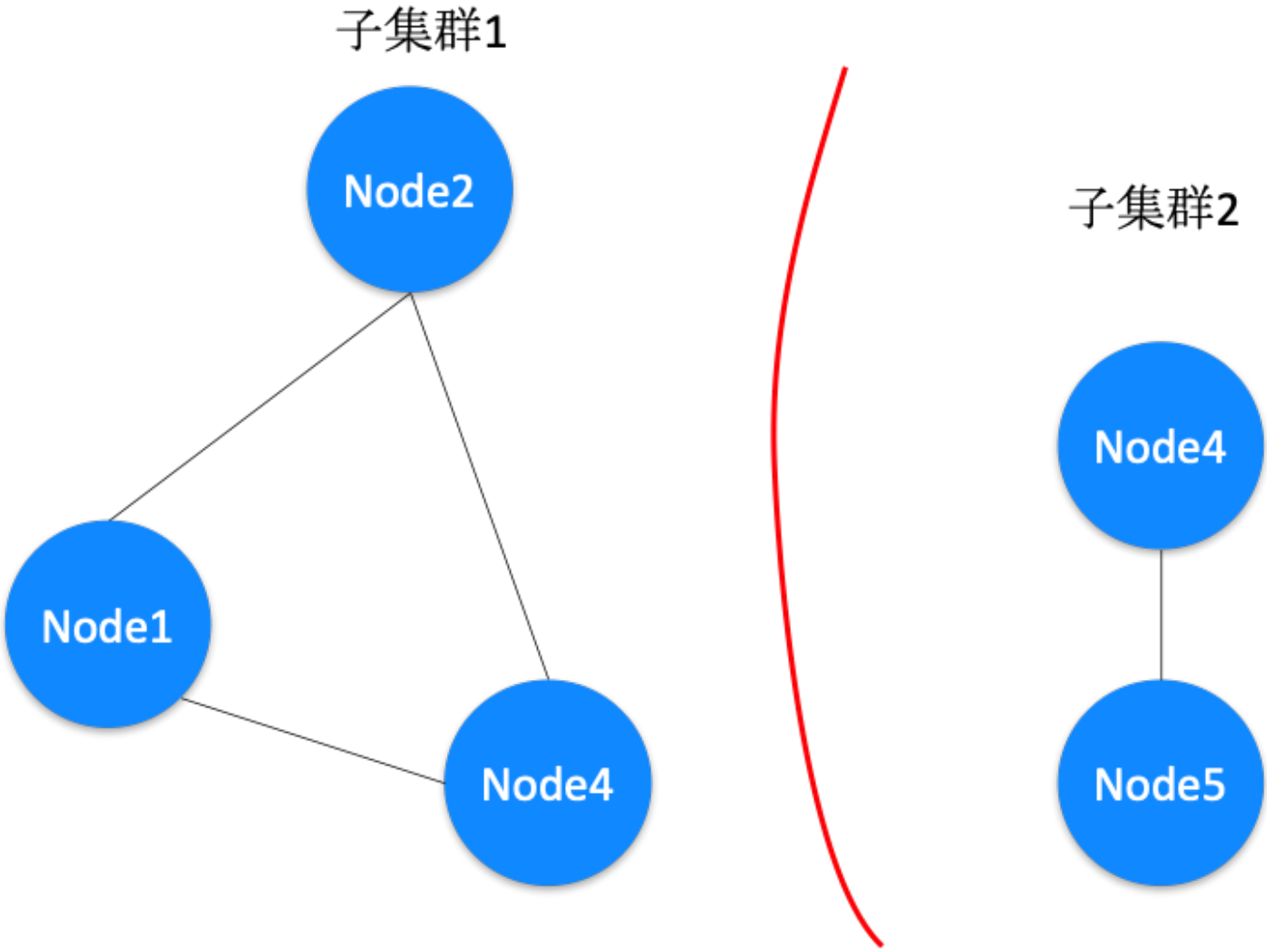

图 2 非集中式架构中网络分区的形态

从集中式和非集中式这两种分布式集群架构的网络分区形态可以看出，**要判断是否形成网络分区，最朴素的方法就是判断节点之间心跳是否超时，然后将心跳可达的节点归属到一个子集中**。

由于非集中式系统中，每个节点都是对等的、提供的服务相同，所以当多个子集群之间不可达，或部分节点出现故障后，尽管提供的服务质量（SLA）可能会下降，但并不影响这些剩余节点或子集群对外提供服务。所以，接下来我将与你重点讨论集中式系统的网络分区问题。

网络分区最微妙的地方在哪里？

在工作和生活中遇到一个问题，你的本能反应估计是，有问题就解决问题好了。而网络分区最微妙的地方在于，你很难通过程序去判断问题到底出在哪里，而只能通过心跳等手段知道部分节点的网络不可达了。

但，导致节点不可达的原因有很多，有可能是网络的原因，也有可能是节点本身的问题。也就是说，我们无法通过一些症状就判断出是否真的产生了分区。另外，你也很难通过程序去判断这个问题是不是很快就会被恢复。这也是应对网络分区问题最微妙的地方。

网络分区出现概率较高的场景是什么？

在第 31 篇文章留言区中有同学提到：

> 个人理解，网络分区故障一般是针对不同集群来说的，单个集群一般在同一个网络中，集群内的单点故障并不会出现网络分区。当整个集群的网络出故障时，才会有分区的说法。能想到检测办法是，单独有机器对不同集群的主节点进行心跳检测来判断。

首先，我要澄清的是，**我们说的网络分区肯定是就同一个集群而言的**。对于不同集群来说，正是因为集群间本就没有太多的交互，才需要从逻辑上分割成不同的集群，这些逻辑上不同的集群本就是可以独立对外提供服务的。

当集群跨多个网络时，确实正如这位同学所说，从概率上讲相对容易出现网络分区的情况，比如一个业务集群部署在多个数据中心时。所以，集群跨多网络部署时，就是网络分区出现概率较高的场景。

接下来，我们看看如何处理网络分区吧。

网络分区有哪些常见的处理方法？

为了不影响分布式系统的高可用性，检测到网络分区后，我们就需要尽快地进行处理。

假如，**我们采用一种非常激进的方式去处理**，即一旦发现节点不可达，则将不可达节点从现有集群中剔除，并在这个新集群中选出新的主。

以图 1 所示集中式集群为例，当备 Master、Slave3 和 Slave4 节点检测到主 Master、Slave1 和 Slave2 节点不可达时，剔除这些不可达节点后，备 Master 升主，连同 Slave3 和 Slave4 节点组成一个新的集群。

如果不可达是由于节点故障导致的，那么这种策略没有任何问题。这些剩余节点组成的集群可以继续对外提供服务。但，如果不可达是因为网络故障引起的，那么集群中的另一个子

集，即主 Master、Slave1 和 Slave2，也会采用同样的策略，仍然对外提供服务。这时，集群就会出现第 22 篇文章中讲到的双主问题了。

假如，**我们采用一种保守的方式去处理**，即节点一旦发现某些节点不可达，则直接停止自己的服务。这样确实解决了双主的问题，但因为不同分区之间的不可达是相互的，且所有的分区都采取了这种停服策略，就会导致系统中所有的节点都停止服务，整个系统完全不可用。这显然也不是我们想看到的。

那么，当系统中出现节点不可达后，如何在不出现双主的情况下，尽可能地提升系统的可用性呢？或者说，有没有什么更均衡的策略呢？

接下来，我就与你分享四种均衡的网络分区处理方法，即 Static Quorum、Keep Majority、设置仲裁机制和基于共享资源的方式。

方法一：通过 Static Quorum 处理网络分区

Static Quorum 是一种固定票数的策略。在系统启动之前，先设置一个固定票数。当发生网络分区后，如果一个分区中的节点数大于等于这个固定票数，则该分区为活动分区。

为了保证发生分区后，不会出现多个活动分区，导致出现双主或多主的问题，需要对固定票数的取值进行一些约束，既：**固定票数≤ 总节点数≤2* 固定票数 - 1**。

这个策略的优点是，简单、容易实现，但却存在两个问题：

一是，对于分区数比较少的时候，比方 2 个分区时，该策略很容易选出一个唯一的活动分区。但是，当活动分区非常多的时候，由于各个分区的票数分散，不容易找到一个满足条件的分区，没有活动分区也就意味着整个集群不可用了。

二是，由于这种策略里固定票数是固定不变的，所以不适用于集群中有动态节点加入的场景。

方法二：通过 Keep Majority 处理网络分区

顾名思义，Keep Majority 就是保留具备大多数节点的子集群。由于不限定每个分区的节点数超过一个固定的票数，所以可以应用于动态节点加入的场景。

假设，集群数为 n，出现网络分区后，保留的子集群为节点数 w≥n/2 的集群。为防止出现双主或两个集群同时工作的情况，通常将集群总节点数 n 设置为奇数。

可想而知，若集群总数为偶数，比如图 1 集中式架构的例子中，子集群 1 和 2 都包含 2 个 Slave 节点，就会导致两个子集群同时存活，在业务场景只允许一个主的情况下，会导致业务运行不正确。

那么，如果真的出现了集群总节点数为偶数，两个子集群节点数均为总数一半时，又应该如何解决分区问题呢?

这时，我们可以在 Keep Majority 的基础上，叠加一些策略，比如保留集群节点 ID 最小的节点所在的子集群。如图 1 所示，假设集群节点总数为 6，现在因为网络故障形成网络分区子集群 1{主 Master，Slave1, Slave2}和子集群 2{备 Master，Slave3, Slave4}，假设 Slave1 是 ID 最小的节点，那么此时要保留包含 Slave1 的子集群 1。

虽然 Keep Majority 方法可以解决动态节点加入的问题，但也不适用于产生多分区的场景。因为随着分区数增多、节点分散，也难以在众多分区中出现一个节点数 w≥n/2 的分区。

前面我讲到集群跨多个网络部署时更容易产生网络分区，因此我不推荐采用 Static Quorum 和 Keep Majority 方法去处理跨多网络集群的网络分区问题。

方法三：通过设置仲裁机制处理网络分区

设置仲裁机制的核心是，引入一个第三方组件或节点作为仲裁者，该仲裁者可以与集群中的所有节点相连接，集群中所有节点将自己的心跳信息上报给这个中心节点。因此，该中心节点拥有全局心跳信息，可以根据全局心跳信息判断出有多少个分区。当出现网络分区后，由仲裁者确定保留哪个子集群，舍弃哪些子集群。

如下图所示，假设引入 Node0 作为第三个节点，该节点 IP 为 10.12.24.35，当出现网络分区子集群 1{Node1, Node3}和子集群 2{Node2, Node4}时，每个子集群中选择一个 Leader 节点并 ping 一下 Node0 的 IP，能 ping 通则保留，否则舍弃。比如下图中，子集群 1 可以 ping 通，子集群 2 ping 不通，因此保留子集群 1。

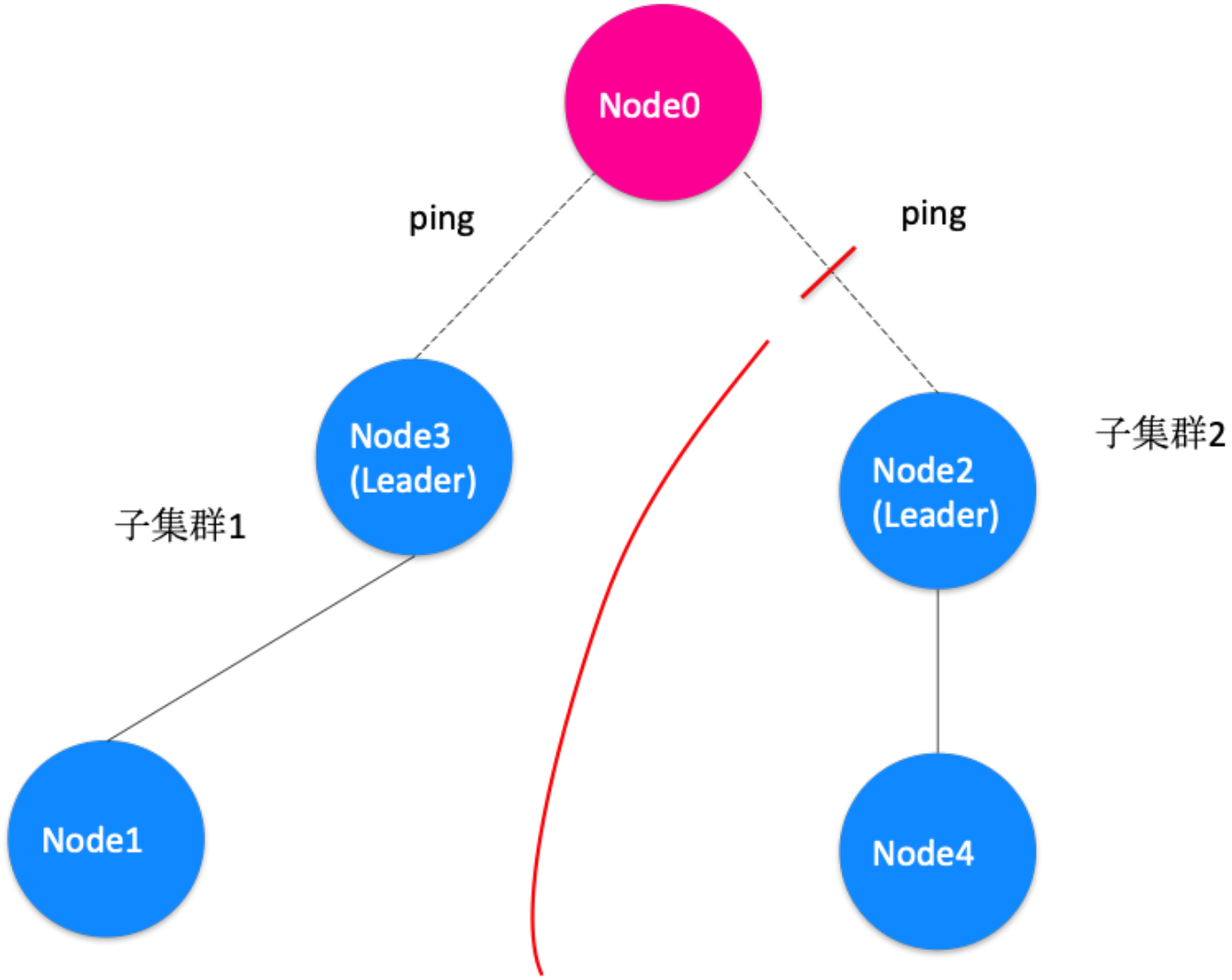

图 3 通过设置仲裁机制处理网络分区

方法四：基于共享资源的方式处理网络分区

说到共享资源，我们需要先回顾下分布式锁的相关知识。分布式锁是实现多个进程有序、避免冲突地访问共享资源的一种方式。

基于共享资源处理网络分区的核心，其实就类似于分布式锁的机制。也就是，哪个子集群获得共享资源的锁，就保留该子集群。获得锁的集群提供服务，只有当该集群释放锁之后，其他集群才可以获取锁。关于锁的管理和获取，你可以再回顾下第 7 篇文章中的相关内容。

这种方式的问题是，如果获取锁的节点发生故障，但未释放锁，会导致其他子集群不可用。因此，这种方式适用于获取锁的节点可靠性有一定保证的场景。

基于仲裁和共享资源的网络分区处理方法，其实都是依赖一个三方的节点或组件，然后借助这个第三方来保证系统中同时只有一个活动分区。所以，这两种处理方法适用于有一个稳定可靠的三方节点或组件的场景。

总结

今天，我与你进一步展开了分布式系统中的网络分区问题，以加深你对网络分区问题的检测、处理方式的理解，并帮助你在实践应用中处理网络分区问题。

关于网络分区的处理方法，其本质就是，在产生分区后，选出一个分区，保证同时最多有一个分区对外提供服务。基于此，我为你梳理了四种常见的处理方法，包括 Static Quorum、Keep Majority、设置仲裁机制和基于共享资源的方式。

其中，基于 Static Quorum 的方法，因为涉及固定票数策略，所以不适用于处理多个分区，以及有动态节点加入的场景；基于 Keep Majority 的方法，可以解决动态节点场景下分区问题，但因为要求子集群节点数≥1/2 总节点数，所以也不适用于处理多个分区的情况；而基于仲裁和共享资源的网络分区处理方法，其实都是依赖一个三方的节点或组件，所以适用于有一个稳定可靠的三方节点或组件的场景。

如果还有哪些思考题或者留言问题，还没来得及扩展的话，你可以留言给我，后续我会再找机会进行解答。最后，我要和你说的是，和我一起打卡分布式核心技术，一起遇见更优秀的自己吧。

篇幅所限，留言区见。

我是聂鹏程，感谢你的收听，欢迎你在评论区给我留言分享你的观点，也欢迎你把这篇文章分享给更多的朋友一起阅读。我们下期再会！

上一篇　31 | 分布式高可用之故障恢复：知错能改，善莫大焉

下一篇　特别放送 | 徐志强：学习这件事儿，不到长城非好汉

精选留言 (5)

写留言

有铭
2019-12-11

引入仲裁者？那仲裁者自己不就是一个薄弱点，仲裁者挂了咋办呢

1

没有昵称
2019-12-12

关于网络分区的处理方法，其本质就是，在产生分区后，选出一个分区，保证同时最多有一个分区对外提供服务

选出一个分区来对外提供服务 其他分区服务停了 感觉有点简单粗暴，如果这个分区扛不住请求压力 又要触发 限流 降级 等一系列操作

展开

Jackey
2019-12-11

之前只考虑了非集中式的分布式系统，没有考虑集中式的。现在有种茅塞顿开的感觉

阿卡牛
2019-12-11

CAP中的网络分区不是指每个分区都可以对外提供服务，但要在数据一致性和可用性间选一个，文稿中说的是只保留一个分区，有点搞不懂

展开

1

leslie
2019-12-11

"网络分区"个人一直觉得是应当算是分布式架构中一个难点：简单的说可能是单独的一小块资源，一个最小集群，可是它只是其中的一部分，例如数据库集群主要就两类，投票或者单独用一台keeplive，可是向上还有一层。

网络分区一定程度上难的应当是嵌套的带来的问题；一套分布式系统里面可能会有多个子系统，例如数据系统、应用系统。。。每个系统内部有又有一套小的，最小模式的网...

展开