**领导人选举，我为什么要关心？**

**更新：**本文以旧名称 Found 引用我们托管的 Elasticsearch 产品。请注意，Found 现在称为 Elastic Cloud。

领导者选举是分布式系统中最棘手的事情之一。同时，理解leader是如何选举产生的以及leader的职责是理解分布式系统的关键。

## [介绍](https://www.elastic.co/cn/blog/found-leader-election-in-general#introduction)

所有分布式系统都必须以一种或另一种方式处理保持一致性。为简化起见，我们可以将策略分为两组。那些试图避免不一致发生的那些，以及那些定义如何协调它们的那些。后者对于适用问题非常强大，但对其他问题的数据模型施加了不可接受的约束。在本文中，我们将研究第一类的一个示例以及它如何应对网络故障。

## [为什么使用领导者？](https://www.elastic.co/cn/blog/found-leader-election-in-general#why-use-a-leader)

在分布式系统中将节点指定为领导者实际上与 Java 中的关键字 synchronized 非常相似。Java 同步的经典示例是两个线程尝试递增相同的整数。例如，当两个线程尝试向同一个帐户分别存入 100 美元时。每个线程都必须读取余额，增加余额并将其写回。给定 100 美元的起始余额并且没有线程同步，可能会发生以下情况：

* 线程 A 读取余额 ($100)
* 线程 B 读取余额 ($100)
* 线程 A 计算新余额 ($200)
* 线程 B 计算新余额 ($200)
* 线程 A 写入新余额 ($200)
* 线程 B 写入新余额 ($200)

很明显，账户从 100 美元开始，总共存入 200 美元，最终余额应该是 300 美元。但是，线程 A 的存款被覆盖，因为线程 B 在写入其更新的余额时已对陈旧的数据进行了计算。

Java 中的解决方案当然是在执行增量的代码上使用关键字 synchronized，它会使一个线程在另一个线程读取余额之前完成其对余额的工作。

如果将线程视为一个节点，很容易想象在分布式系统中出现同样的问题，但没有 synchronized 关键字可以解决问题。原因是同步是使用信号量实现的，反过来，它们由底层操作系统和运行它的硬件支持。但是，如果我们看看 synchronized 做了什么，我们可能会将这个概念应用于分布式设置。在 Java 中将代码块声明为同步时，它总是同步到特定对象的监视器。这台显示器的尺寸是一。这意味着在任何给定时间只有一个线程可能在监视器内。当一个线程请求监视器并且监视器可用时，它被允许立即进入。如果监视器被占用，则线程被放入等待池并挂起，直到监视器被释放。

我们如何使其适应分布式环境？好吧，我们确实可以选择在每个节点上创建任意数量的监视器，但是对于需要保护的给定资源，可能只存在一个监视器。换句话说，每个账户余额可能只有一个监视器。这将维护数据一致性的问题转化为确保所有节点就哪个节点负责为每个余额实施监控的问题达成一致。在我们的简单示例中，这可以由领导者处理，剩下要做的就是选举领导者。

如果你有像我这样的点对点背景，你可能会建议使用分布式哈希表而不是领导者来决定哪个节点为每个资源实现监视器。事实上，有一些非常棒的 DHT 实现，每小时处理数千个节点的离开和加入。鉴于它们也可以在没有底层网络拓扑先验知识的异构网络中工作，四到十个消息跳的查询响应时间相当不错，但是在节点同质且网络相当稳定的网格设置中，我们可以做得更好。

Elasticsearch 典型场景中的另一个简化是集群中没有那么多节点。通常，节点的数量远远少于单个节点能够维持的连接数量，并且网格环境不必处理频繁加入和离开的节点。这就是领导者方法更适合 Elasticsearch 的原因。

## [禅道](https://www.elastic.co/cn/blog/found-leader-election-in-general#the-zen-way)

所以我们有一个领导者，定期将当前版本的全局集群状态推送到每个节点。如果领导倒下了怎么办？仅仅因为我们的节点相当可靠并不意味着我们可以接受单点故障。假设只有领导节点崩溃并且所有其他节点仍然能够通信，Elasticsearch 将优雅地处理此问题，因为其余节点将检测到领导者的故障 - 或者更确切地说，领导者没有消息 - 并启动领导者选举。如果您使用 ZenDiscovery（默认），则过程如下：

* 每个节点计算已知的最小节点 id 并向该节点发送领导投票
* 如果一个节点收到足够多的选票，并且该节点也为自己投票，那么它就会扮演领导者的角色并开始发布集群状态。

The definition of sufficiently many votes to win an election is what is known as a quorum. 在 Elasticsearch 中，仲裁大小是一个可配置的参数。分布式系统中的节点无法确定另一个节点是否已死亡或网络是否无法将其消息传递给它，因此通常有一个预先商定的阈值 - 法定人数 - 来代表另一个节点党的选票。让我们以以下场景为例：集群有节点 A、B、C、D、E 和 quorum 大小为 3。A 是领导者。碰巧 A 和 B 在一个网络上，而 C、D 和 E 在另一个网络上。网络通过链路连接。

初始网络

当这条链路发生故障时，A 和 B 仍然可以相互通信，但不能与 C、D 和 E 通信。同样，在另一个网络上 C、D 和 E 可以相互通信，但不能与 A 和 B 通信。

分区网络

接下来发生的事情是：节点 C、D 和 E 检测到他们不再与领导者 A 联系，然后通过向 C 发送投票来发起领导者选举。一旦 C 收到三张选票，它就会扮演领导者的角色并开始发布到 C、D 和 E。在另一个网络上，领导者 A 检测到它不再与节点 C、D 和 E 联系。领导者 A 计算出新的集群大小小于仲裁大小并放弃领导者角色并有效地阻止节点 A 和 B 响应查询，直到链接恢复。

在现实生活中，不太可能有人在关键的网络电缆上绊倒，但网络比我上面的示例更复杂，网络分区实际上并不少见。当一个系统依赖多个数据中心时，不难想象网络分裂；棘手的网络错误的另一个可能罪魁祸首是路由器或交换机配置错误。正如在[**网络可靠**](http://aphyr.com/posts/288-the-network-is-reliable)中提到的那样，确实会发生网卡在传递出站数据包的同时丢弃所有入站数据包的事情，结果服务器仍然发送心跳但无法为任何请求提供服务。

## [避免脑裂](https://www.elastic.co/cn/blog/found-leader-election-in-general#avoid-split-brain)

仲裁大小的概念有两个直接的优点。首先，它简化了选举过程，因为只需将选票传递给他们投票的节点。其次，更重要的是，这是避免脑裂的唯一方法。想象一下同一个集群和同一个网络分裂，但仲裁大小为 2。C 仍将被选为领导者，但 A 不会放弃其领导者角色。这将导致两个领导者，彼此不知道集群。这就是所谓的裂脑。两个集群都将接受读写操作，并且正如预期的那样，它们将不同步。如果没有人为干预，他们可能永远无法恢复。根据数据模型，可能无法统一两个数据版本，迫使其中一个简单地丢弃其中一个集群上的所有数据。

## [不要重新发明轮子](https://www.elastic.co/cn/blog/found-leader-election-in-general#dont-reinvent-the-wheel)

正如你所料，领导人选举多年来一直是学术界的一个有趣话题，不少聪明人都进行了很多思考。如果您更热衷于让您的分布式系统运行而不是投入大量精力进行研究和开发，那么您最好尝试实现一个众所周知的算法。在未来的某一天，当您最终发现系统中的疯狂错误时，它更有可能只是实现中的错误，而不是算法中的主要设计缺陷。当然，同样的论点适用于调整或集成现有实现，而不是从头开始实现，特别是如果您想要更高级的算法之一。[**此错误报告**](https://github.com/elasticsearch/elasticsearch/issues/2488)说明了创建一个好的领导选举实现是多么棘手，即使你有一个健全的算法。

### [恶霸算法](https://www.elastic.co/cn/blog/found-leader-election-in-general#the-bully-algorithm)

欺凌算法是leader选举的基本算法之一。它假定所有节点都被赋予了一个唯一的 ID，该 ID 对节点进行了总排序。任何时候的当前leader都是参与集群的id最高的节点。该算法的优点是易于实现，但它不能很好地应对具有最大 id 的节点不稳定的情况。尤其是在具有最大 id 的节点往往会因领导角色的杂务而过载的情况下。因此，它将作为领导者崩溃；id第二大的节点将被选举；并且最大的 id 节点将恢复 - 因为它不再过载 - 随后再次启动领导者选举，只是被选举并再次崩溃。然而，在低级硬件接口中，欺凌算法相当普遍。

### [Paxos](https://www.elastic.co/cn/blog/found-leader-election-in-general#paxos)

Paxos 实际上不仅仅是一个领导者选举算法。Paxos 是一系列不同的协议，用于在分布式系统中维护共识。我不会详细介绍 Paxos 的变体，而是大致讨论 Paxos 的概念和质量。Paxos 中使用的数据模型是一个不断增长的语句列表，其中每个语句都有一个唯一的编号。Paxos 的数学证明保证是两个节点永远不会对相同的数字有不同的语句，并且只要有一定数量的节点参与，算法就会取得进展。这意味着节点永远不会不一致，算法永远不会陷入死锁，但如果没有足够的节点在线，它可能会停止。这些保证实际上与我们想要的领导人选举算法非常相似。我们希望参与集群的所有节点就哪个节点是领导者达成一致，并且我们不希望任何其他节点能够跑掉并创建自己的集群，从而导致脑裂。然后，使用 Paxos 执行领导者选举就像提出“我是领导者”这样的声明一样简单。如果它被接受，那么你就是领导者，直到另一个节点提议成为领导者。然而，仅此一点还不足以避免脑裂。在上面的分区网络示例中，现有领导者 A 不会被通知节点 C 在另一台交换机上的选举。需要有另一个标准来结束领导。一种选择是，在发起领导人选举时，使用的语句是这样的：“我是领导人，直到 而且我们不希望任何其他节点能够运行并创建自己的集群，从而导致脑裂。然后，使用 Paxos 执行领导者选举就像提出“我是领导者”这样的声明一样简单。如果它被接受，那么你就是领导者，直到另一个节点提议成为领导者。然而，仅此一点还不足以避免脑裂。在上面的分区网络示例中，现有领导者 A 不会被通知节点 C 在另一台交换机上的选举。需要有另一个标准来结束领导。一种选择是，在发起领导人选举时，使用的语句是这样的：“我是领导人，直到 而且我们不希望任何其他节点能够运行并创建自己的集群，从而导致脑裂。然后，使用 Paxos 执行领导者选举就像提出“我是领导者”这样的声明一样简单。如果它被接受，那么你就是领导者，直到另一个节点提议成为领导者。然而，仅此一点还不足以避免脑裂。在上面的分区网络示例中，现有领导者 A 不会被通知节点 C 在另一台交换机上的选举。需要有另一个标准来结束领导。一种选择是，在发起领导人选举时，使用的语句是这样的：“我是领导人，直到 如果它被接受，那么你就是领导者，直到另一个节点提议成为领导者。然而，仅此一点还不足以避免脑裂。在上面的分区网络示例中，现有领导者 A 不会被通知节点 C 在另一台交换机上的选举。需要有另一个标准来结束领导。一种选择是，在发起领导人选举时，使用的语句是这样的：“我是领导人，直到 如果它被接受，那么你就是领导者，直到另一个节点提议成为领导者。然而，仅此一点还不足以避免脑裂。在上面的分区网络示例中，现有领导者 A 不会被通知节点 C 在另一台交换机上的选举。需要有另一个标准来结束领导。一种选择是，在发起领导人选举时，使用的语句是这样的：“我是领导人，直到”，但更优选的选择是领导者能够检测到它不再与法定人数有联系，然后将自己降级。

鉴于前面的网络分区示例，让我们假设管理员在连接到节点 C 的网络电缆上跳闸，然后将电缆重新连接到另一个交换机，从而导致网络的新分区：

移动节点

有趣的是，quorum 现在由节点 A、B 和 C 组成，而不是之前的 C、D 和 E。如果运行 Paxos 的节点，它们的数据可能如下所示：

Paxos 示例

根据通过 Paxos 进行领导者选举的具体实现，可能会出现许多结果，但它们都有两个共同的属性。节点 D 和 E 将无法选举新的领导者，因为他们没有多数。节点 A 和 B 在得知节点 C 在第 2 轮中被选举之前不会选举新的领导者。假设每个领导者任期的结束标准是超时，有两种可能的结果。如果节点 C 在其任期结束前与节点 A 和 B 建立联系，它将重新建立仲裁，恢复其领导者角色，无需进行领导者选举。如果其任期结束，则可以选举节点 A、B 或 C。假设节点A是第一个在网络上发现节点C的节点C，它将尝试重新选举为第2术语的领导者，但节点C将拒绝这一点，因为它已经有一个术语2的领导者。节点A可以选择尝试选举学期3.如果它确实，它也会向节点B通知节点C的时间作为第2期的领导者。现在，如果节点C和试图选择什么同时？这个问题的完整答案需要深入研究 Paxos 的细节，这超出了本文的范围，但简短的回答是选举会失败。实际上，最初的 paxos 论文建议使用领导者选举算法来决定哪个节点应该是发起新提案的节点，但是由于 Paxos 协议保证即使有多个节点提出冲突的声明也不会出现不一致，所以这次领导者选举算法可能只是一个简单的启发式算法，以避免连续失败的回合。例如，一个节点可能会决定等待一段随机的时间，然后再重新尝试提出一个失败的语句。Paxos 在何时以及如何进行领导者选举方面的这种灵活性可能比简单的欺负算法具有很大的优势，因为在现实生活中，故障模式比死网络链接要多得多。

## [结论](https://www.elastic.co/cn/blog/found-leader-election-in-general#conclusion)

在这篇文章中，我试图简要介绍学术界在领导人选举中的一些工作以及正确行事的重要性。精简版是这样的：

* 如果您关心您的数据，则法定人数非常重要。
* Paxos 确实很健壮，但实现起来并不容易。
* 最好实现一个众所周知的算法，其中优势和缺陷是已知的，而不是在未来得到一个大惊喜。当然有很多选择。