Raft 算法（3）：如何解决成员变更

jask

2024-08-11

Raft 如何解决集群配置变更时，集群能稳定运行，不出现两个领导者的呢?

成员变更时，最大的风险是可能出现两个领导者：

比如在进行成员变更时，节点 A、B、C 之间发生了分区错误，节点 A、B 组成旧配置中的“大多数”，也就是变更前的 3 节点集群中的“大多数”，那么这时的领导者（节点 A）依旧是领导者。

另一方面，节点 C 和新节点 D、E 组成了新配置的“大多数”，也就是变更后的 5 节点集群中的“大多数”，它们可能会选举出新的领导者（比如节点 C）。那么这时，就出现了同时存在 2 个领导者的情况。

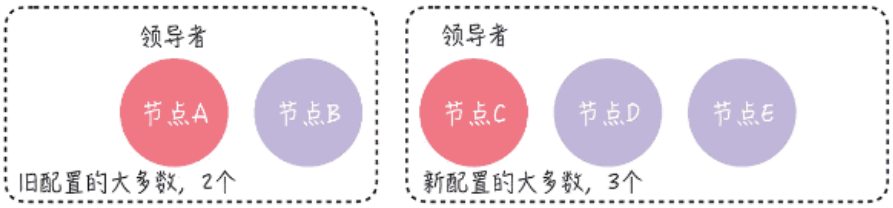


Figure 1: 情况

解决方法：单节点变更

单节点变更，就是通过一次变更一个节点实现成员变更。如果需要变更多个节点，那你需要执行多次单节点变更。比如将 3 节点集群扩容为 5 节点集群，这时你需要执行 2 次单节点变更，先将 3 节点集群变更为 4 节点集群，然后再将 4 节点集群变更为 5 节点集群。

过程

假定 A 是领导者：
目前的集群配置为 [A, B, C]，我们先向集群中加入节点 D，这意味着新配置为 [A, B, C, D]。成员变更，是通过这么两步实现的：

第一步，领导者（节点 A）向新节点（节点 D）同步数据；
第二步，领导者（节点 A）将新配置 [A, B, C, D] 作为一个日志项，复制到新配置中所有节点（节点 A、B、C、D）上，然后将新配置的日志项提交到本地状态机，完成单节点变更。
新的节点同理。

在正常情况下，不管旧的集群配置是怎么组成的，旧配置的“大多数”和新配置的“大多数”都会有一个节点是重叠的。也就是说，不会同时存在旧配置和新配置 2 个“大多数”：

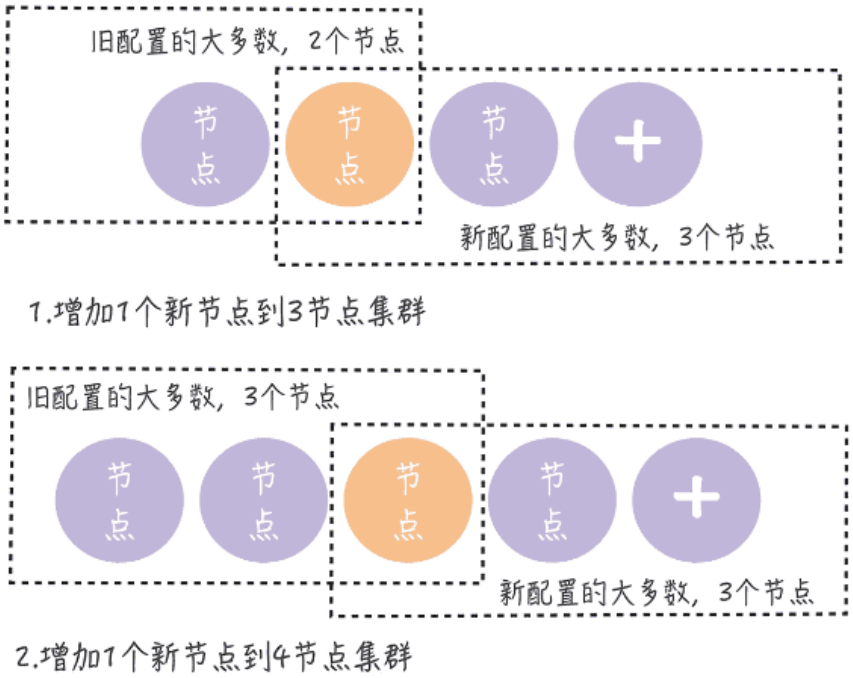


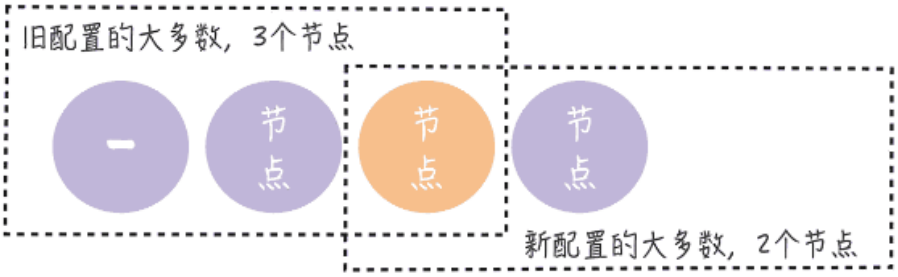
Figure 2: 示意

分区错误、节点故障等情况下，如果我们并发执行单节点变更，那么就可能出现一次单节点变更尚未完成，新的单节点变更又在执行，导致集群出现 2 个领导者的情况。

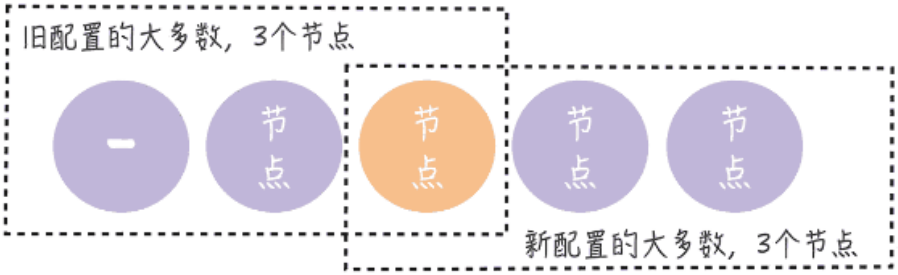
如果你遇到这种情况，可以在领导者启动时，创建一个 NO_OP 日志项（也就是空日志项），只有当领导者将 NO_OP 日志项提交后，再执行成员变更请求。这个解决办法，你记住就可以了，可以自己在课后试着研究下。

总结

成员变更的问题，主要在于进行成员变更时，可能存在新旧配置的 2 个“大多数”，导致集群中同时出现两个领导者，破坏了 Raft 的领导者的唯一性原则，影响了集群的稳定运行。单节点变更是利用“一次变更一个节点，不会同时存在旧配置和新配置 2 个‘大多数’”的特性，实现成员变更。因为联合共识实现起来复杂，不好实现，所以绝大多数 Raft 算法的实现，采用的都是单节



3.从4节点集群中移除1个节点



4.从5节点集群中移除1个节点

Figure 3: 示意

点变更的方法（比如 Etcd、Hashicorp Raft）。其中，Hashicorp Raft 单节点变更的实现，是由 Raft 算法的作者迭戈·安加罗（Diego Ongaro）设计的，很有参考价值。

其实 Raft 不是一致性算法而是共识算法，是一个 Multi-Paxos 算法，实现的是如何就一系列值达成共识。并且，Raft 能容忍少数节点的故障。虽然 Raft 算法能实现强一致性，也就是线性一致性（Linearizability），但需要客户端协议的配合。在实际场景中，我们一般需要根据场景特点，在一致性强度和实现复杂度之间进行权衡。比如 Consul 实现了三种一致性模型。

default：客户端访问领导者节点执行读操作，领导者确认自己处于稳定状态时（在 leader leasing 时间内），返回本地数据给客户端，否则返回错误给客户端。在这种情况下，客户端是可能读到旧数据的，比如此时发生了网络分区错误，新领导者已经更新过数据，但因为网络故障，旧领导者未更新数据也未退位，仍处于稳定状态。

consistent：客户端访问领导者节点执行读操作，领导者在和大多数节点确认自己仍是领导者之后返回本地数据给客户端，否则返回错误给客户端。在这种情况下，客户端读到的都是最新数据。

stale：从任意节点读数据，不局限于领导者节点，客户端可能会读到旧数据。