**ES选举类Bully算法**

Bully算法

bully算法是一个分布式系统中动态选择master节点的算法，进程号最大的非失效的节点将被选为master。  
算法用三种消息类型：

1）选举消息 （Election Message: Sent to announce election.）

2）应答消息（Answer (Alive) Message: Responds to the Election message.）

3）选举成功消息 （Coordinator (Victory) Message: Sent by winner of the election to announce victory.）

当一个进程P从失败中恢复，或者接收到主节点失效信息，进程P将做以下事情：

1）如果进程P有最大的进程ID，那么它则会向其他节点广播Coordinator (Victory) Message。否则进程P向进程号比它大的进程发送Election Message

2）如果进程P发送Election Message后，没有接收到应答，它就会向其他节点广播Coordinator (Victory) Message，并成为master。

3）如果进程P接收到比它进程号更高的进程的Answer (Alive) Message信息，那么它这次的选举就失败了，等待接收其他节点的Coordinator (Victory) Message。

4）如果进程P接收到比它进程号低的进程的Election message，那么它会向该节点返回一个Answer (Alive) Message，并启动选举进程，并向比它更高的进程发送Election Message。

5）如果进程P接收到Coordinator (Victory) Message，那么它就会把发送这条消息的节点看作为master进程。

ES Master选举过程

我看的源码是5.6版本的。因此以下的解释都是依据5.6的源码来说的。  
当master节点失效之后，各个节点find master的过程如下：  
1）每个节点会ping配置文件中discovery.zen.ping.unicast.hosts的IP，找到所有存活的节点并过滤  
2）找到非本身的active master

List<DiscoveryNode> activeMasters = new ArrayList<>();

for (ZenPing.PingResponse pingResponse : pingResponses) {

// We can't include the local node in pingMasters list, otherwise we may up electing ourselves without

// any check / verifications from other nodes in ZenDiscover#innerJoinCluster()

if (pingResponse.master() != null && !localNode.equals(pingResponse.master())) {

activeMasters.add(pingResponse.master());

}

}

3）找到所有的可成为master的节点集合masterCandidates ，包含自己

// nodes discovered during pinging

List<ElectMasterService.MasterCandidate> masterCandidates = new ArrayList<>();

for (ZenPing.PingResponse pingResponse : pingResponses) {

if (pingResponse.node().isMasterNode()) {

masterCandidates.add(new ElectMasterService.MasterCandidate(pingResponse.node(), pingResponse.getClusterStateVersion()));

}

}

4）如果activeMasters 为空，也就是说不存在活着的master节点，同时当前活着的节点满足配置中discovery.zen.minimum\_master\_nodes的数量，那么就从masterCandidates 挑出ID最小的节点，让其成为master节点。如果activeMasters 不为空，则从中选择最小的ID成为Master节点即可。

if (activeMasters.isEmpty()) {

if (electMaster.hasEnoughCandidates(masterCandidates)) {

final ElectMasterService.MasterCandidate winner = electMaster.electMaster(masterCandidates);

logger.trace("candidate {} won election", winner);

return winner.getNode();

} else {

// if we don't have enough master nodes, we bail, because there are not enough master to elect from

logger.warn("not enough master nodes discovered during pinging (found [{}], but needed [{}]), pinging again",

masterCandidates, electMaster.minimumMasterNodes());

return null;

}

} else {

assert !activeMasters.contains(localNode) : "local node should never be elected as master when other nodes indicate an active master";

// lets tie break between discovered nodes

return electMaster.tieBreakActiveMasters(activeMasters);

}

electMaster.electMaster方法和electMaster.tieBreakActiveMasters方法则都是从集合中选取最小节点的ID：

public MasterCandidate electMaster(Collection<MasterCandidate> candidates) {

assert hasEnoughCandidates(candidates);

List<MasterCandidate> sortedCandidates = new ArrayList<>(candidates);

sortedCandidates.sort(MasterCandidate::compare);

return sortedCandidates.get(0);

}

public DiscoveryNode tieBreakActiveMasters(Collection<DiscoveryNode> activeMasters) {

return activeMasters.stream().min(ElectMasterService::compareNodes).get();

}

如果当前不存在active master，那么activeMasters 一定为空，则从masterCandidates 从选出id最小的节点即可。  
如果当前存在active master，且当前节点不是active maste，那么从activeMasters 中选出id最小的节点。  
如果当前存在active master，且当前节点是active maste，那么activeMasters 为空，从masterCandidates 中选出id最小的节点即自己。

在我的感觉中，当前active master的个数要么为空，要么为1，这边不知道为