图解zookeeper FastLeader选举算法

Zookeeper 算法 2014-10-20 20:32:01 发布

zookeeper配置为集群模式时，在启动或异常情况时会选举出一个实例作为Leader。其默认选举算法为 `FastLeaderElection` 。

不知道zookeeper的可以考虑这样一个问题：某个服务可以配置为多个实例共同构成一个集群对外提供服务。其每一个实例本地都存有冗余数据，每 一个实例都可以直接对外提供读写服务。在这个集群中为了保证数据的一致性，需要有一个Leader来协调一些事务。那么问题来了：如何确定哪一个实例是 Leader呢？

问题的难点在于：

- 没有一个仲裁者来选定Leader
- 每一个实例本地可能已经存在数据，不确定哪个实例上的数据是最新的

分布式选举算法正是用来解决这个问题的。

本文基于zookeeper 3.4.6 的源码进行分析。FastLeaderElection算法的源码全部位于 `FastLeaderElection.java` 文件中，其对外接口为 `FastLeaderElection.lookForLeader` ，该接口是一个同步接口，直到选举结束才会返回。同样由于网上已有类似文章，所以我就从图示的角度来阐述。阅读一些其他文章有利于获得初步印象：

- 深入浅出Zookeeper之五 Leader选举，代码导读
- zookeeper3.3.3源码分析(二)FastLeader选举算法，文字描述较细

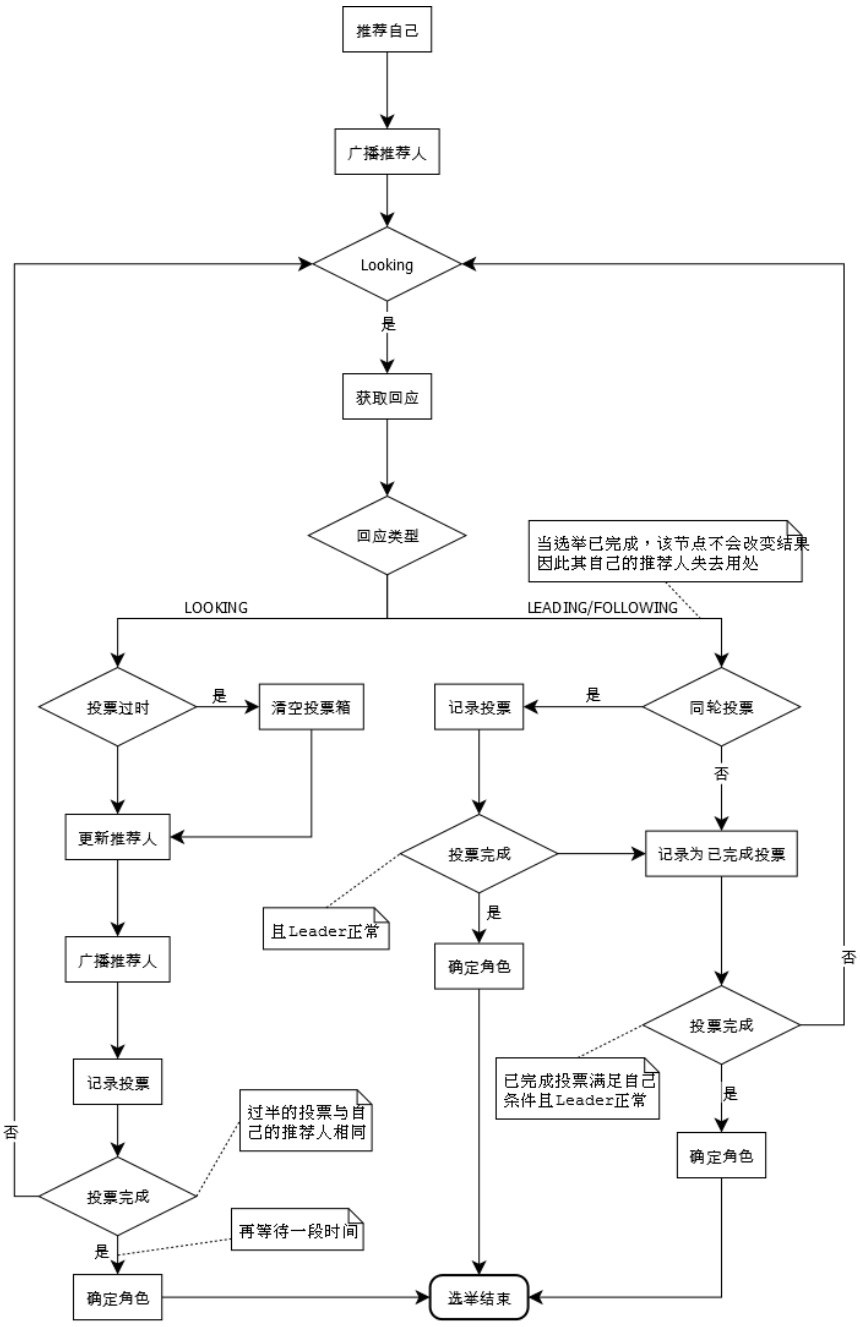
主要流程

阅读代码和以上推荐文章可以把整个流程梳理清楚。实现上，包括了一个消息处理主循环，也是选举的主要逻辑，以及一个消息发送队列处理线程和消息解码线程。主要流程可概括为下图：

同类热门经验

1. 一文读懂机器学习，大数
语言处理/算法全有了…
2. 各种Java加密算法
3. C语言的学习基础,100个
算法
4. MapReduce 模式、算法
（MapReduce Patterns
Algorithms, and Use Ca
5. Mahout算法集
6. Java RSA 加密解密算法

阅读目录



推荐对照着推荐的文章及代码理解，不赘述。

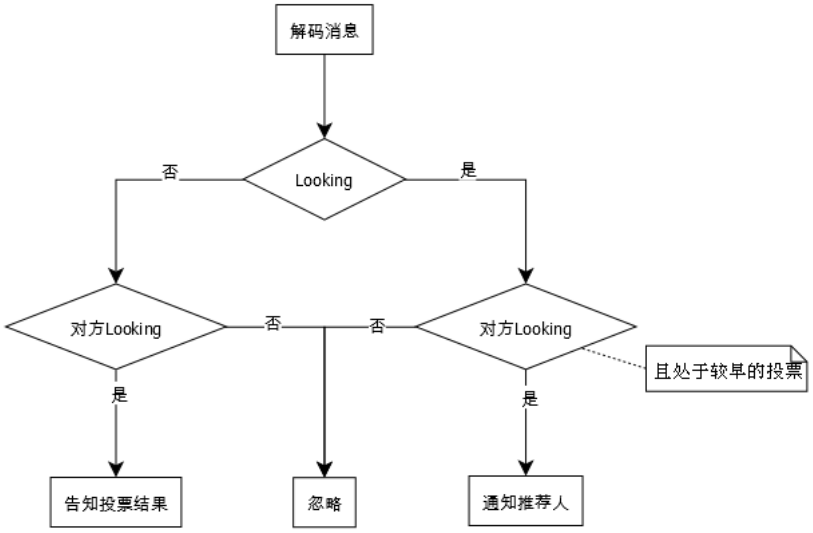
我们从感性上来理解这个算法。

每一个节点，相当于一个选民，他们都有自己的推荐人，最开始他们都推荐自己。谁更适合成为Leader有一个简单的规则，例如sid够大（配置）、 持有的数据够新(zxid够大)。每个选民都告诉其他选民自己目前的推荐人是谁，类似于出去搞宣传拉拢其他选民。每一个选民发现有比自己更适合的人时就转 而推荐这个更适合的人。最后，大部分人意见一致时，就可以结束选举。

就这么简单。总体上有一种不断演化逼近结果的感觉。

当然，会有些特殊情况的处理。例如总共3个选民，1和2已经确定3是Leader，但3还不知情，此时就走入 `LEADING/FOLLOWING` 的分支，选民3只是接收结果。

代码中不是所有逻辑都在这个大流程中完成的。在接收消息线程中，还可能单独地回应某个节点（ `WorkerReceiver.run` ）：



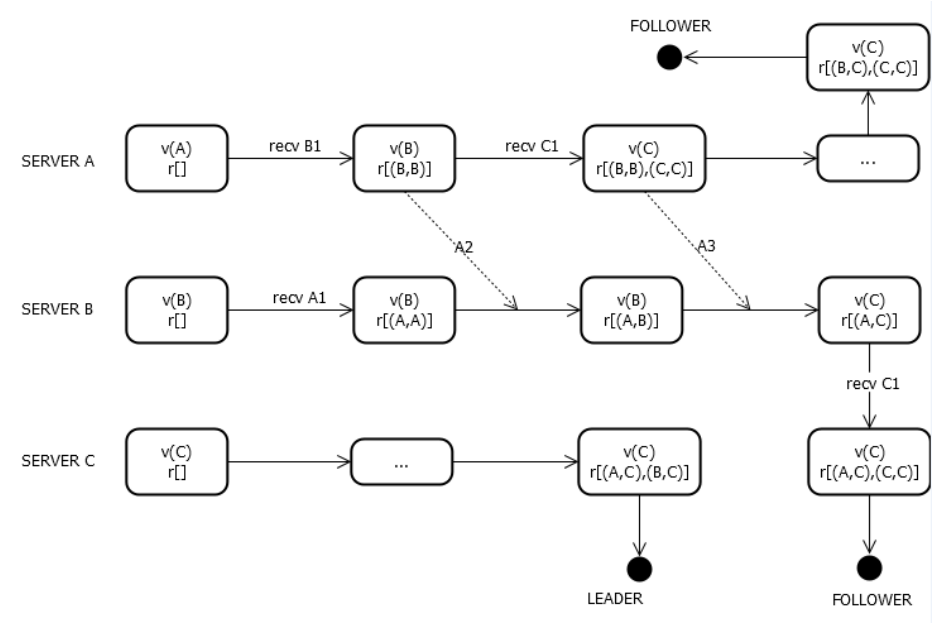
从这里可以看出，当某个节点已经确定选举结果不再处于 `LOOKING` 状态时，其收到 `LOOKING` 消息时都会直接回应选举的最终结果。结合上面那个比方，相当于某次选举结束了，这个时候来了选民4又发起一次新的选举，那么其他选民就直接告诉它当前的Leader情况。相当于，在这个集群主从已经就绪的情况下，又开启了一个实例，这个实例就会直接使用当前的选举结果。

状态转换

每个节点上有一些关键的数据结构：

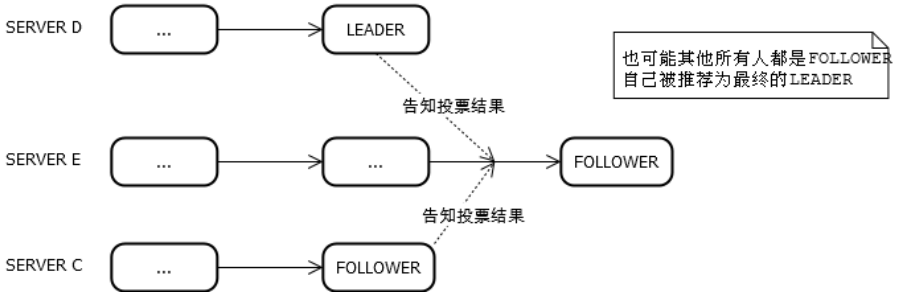
- 当前推荐人，初始推荐自己，每次收到其他更好的推荐人时就更新
- 其他人的投票集合，用于确定何时选举结束

每次推荐人更新时就会进行广播，正是这个不断地广播驱动整个算法趋向于结果。假设有3个节点A/B/C，其都还没有数据，按照sid关系为C>B>A，那么按照规则，C更可能成为Leader，其各个节点的状态转换为：



图中，v(A)表示当前推荐人为A；r[]表示收到的投票集合。

可以看看当其他节点已经确定投票结果时，即不再是 `LOOKING` 时的状态：



代码中有一个特殊的投票集合 `outofelection` ，我理解为选举已结束的那些投票，这些投票仅用于表征选举结果。

当一个新启动的节点加入集群时，它对集群内其他节点发出投票请求，而其他节点已不处于 `LOOKING` 状态，此时其他节点回应选举结果，该节点收集这些结果到 `outofelection` 中，最终在收到合法LEADER消息且这些选票也构成选举结束条件时，该节点就结束自己的选举行为。注意到代码中会 `logicalclock = n.electionEpoch;` 更新选举轮数

完

> 原文地址： http://codemacro.com/2014/10/19/zk-fastleaderelection/
> written by Kevin Lynx posted athttp://codemacro.com