不做过多重点和面试出现问题的解释，全篇内容均为重点！切记！你可以了解到如下面试知识点：

（1）为什么在Zookeeper中Server 数目一般为奇数？

（2）为什么要进行Leader选举？

（3）什么时候会进行Leader的选举？

（4）Leader选举的条件是什么？

（5）选举过程中集群节点的状态是什么？

（6）Leader的选择机制的方式有哪几种？常用的是哪一个？

（7）SID、ZXID、投票、选举状态、Quorum等关键概念的含义？

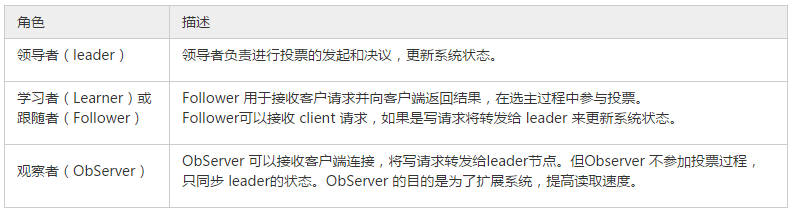
（8）服务器启动时期的Leader选举过程是什么？

（9）服务器运行时期的Leader选举过程是什么？

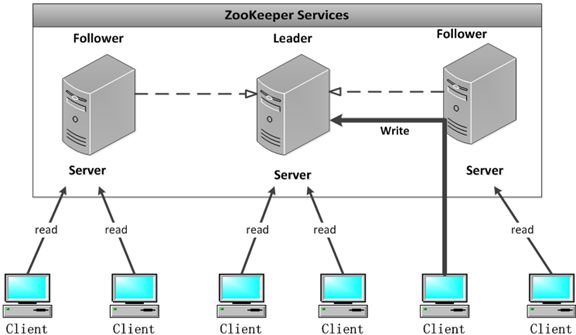
（10）选举算法的核心是什么，根据什么样的规则来进行选举Leader的？

# 一、为什么要进行Leader选举？

配置多个实例共同构成一个集群对外提供服务以达到水平扩展的目的，每个服务器上的数据是相同的，每一个服务器均可以对外提供读和写的服务，对客户端来讲每个服务器都是平等的。为了进一步提高集群的性能，Zookeeper将集群中机器分为了三种角色:Leader、Follower、Observer。



* 集群中将选举出一个Leader，其他的机器则称为Follower，所有的写操作都被传送给Leader，并通过brodcast将所有的更新告诉给Follower
* Leader主要作用是保证分布式数据一致性，即每个节点的存储的数据同步，Leader选举是保证分布式数据一致性的关键所在。
* 当Leader崩溃或者Leader失去大多数的Follower时，需要重新选举出一个新的Leader，让所有的服务器都恢复到一个正确的状态。



**1、Leader选举分为两种情况：**

* Zookeeper集群初始化启动时选举；
* Zookeeper集群运行期间Leader重新选举。

**2、Leader选举的条件：**

和上边的情况是一样的，只不过描述的角度稍微不一样。

* 服务器初始化启动；
* 服务器运行期间无法和Leader保持连接；
* Leader节点崩溃，逻辑时钟崩溃；

**3、集群节点的状态**

当一台机器进入Leader选举流程的时候，当前集群也可能会处于以下两种状态：

* 集群中本来就已经存在一个Leader；
* 集群中确实不存在Leader；

上述第一种情况已经存在Leader的情况，这种情况通常是集群中的某一台机器启动的比较晚，在它启动之前，集群已经可以正常的工作，即已经存在了一台Leader服务器。针对这种情况，当该机器尝试去选举Leader的时候，就会被告知当前服务器的Leader信息，对于该机器来说，仅仅需要和Leader机器建立起来连接，并进行状态的同步即可。

# 二 、Leader的选择机制的方式

Zookeeper提供了三种方式：

* LeaderElection
* TCP版本的FastLeaderElection
* UDP版本的FastLeaderElection，又分为授权模式和非授权模式；

关于这几种选举的方式是可以通过配置文件进行配置的，从3.4.0版本开始，Zookeeper只保留了TCP版本的FastLeaderElection选举机制。所以这篇主要分析它的选举机制。

# 三、选择机制中的概念

**1、服务器ID：SID**

比如有三台服务器，编号分别是1,2,3。编号越大在选择算法中的权重越大。

这个值是myid，也是SID，即SID=myid，这个myid使我们在搭建Zookeeper集群的时候指定的数字编号。

**2、数据ID**

服务器中存放的最大数据ID。值越大说明数据越新，在选举算法中数据越新权重越大。

**3、逻辑时钟**

或者叫投票的次数，用来判断多个投票是否在同一轮选举周期中，同一轮投票过程中的逻辑时钟值是相同的。每投完一次票这个数据就会增加，然后与接收到的其它服务器返回的投票信息中的数值相比，根据不同的值做出不同的判断。

**4、选举状态**

* LOOKING，寻找Leader状态。当服务器处于该状态时，它会认为当前集群中没有Leader，会像集群中所有其他消息发送消息，因此需要进入Leader选举状态。
* FOLLOWING，随从状态，同步leader状态，参与投票。表明当前服务器角色是Follower。
* OBSERVING，观察状态，同步leader状态，不参与投票。表明当前服务器角色是Observer。
* LEADING，领导者状态。表明当前服务器角色是Leader。

**5、ZXID:事务ID**

被推举的Leader事务ID，用来唯一标识一次服务器状态的变更。在某一时刻，集群中每台机器的ZXID值不一定全都一样。也可以看成是服务器中存放的最新数据version，值越大说明数据越新，在选举中数据越新权重越大。

**6、Vote：投票**

Leader选举通过投票实现。当满足选举条件的时候就会开始进行投票。

**7、Quorum：过半机器数**

Quorum是整个Leader选举算法中最为重要的一个术语，指的是Zookeeper汲取求求你中国版的机器数，如果集群中有n台机器，那个quorum的值为：



# 四、服务器启动时期的Leader选举

若进行Leader选举，则至少需要两台机器，这里选取3台机器组成的服务器集群为例。在集群初始化阶段，当有一台服务器Server1启动时，其单独无法进行和完成Leader选举，当第二台服务器Server2启动时，此时两台机器可以相互通信，每台机器都试图找到Leader，于是进入Leader选举过程。

选举过程如下：

1、**每个Server发出一个投票**。由于是初始情况，Server1和Server2都会将自己作为Leader服务器来进行投票，每次投票会包含所推举的服务器的SID和ZXID，使用(SID, ZXID)来表示，此时Server1的投票为(1, 0)，Server2的投票为(2, 0)，然后各自将这个投票发给集群中其他机器。

2、**接受来自各个服务器的投票**。集群的每个服务器收到投票后，首先判断该投票的有效性，如检查是否是本轮投票、是否来自LOOKING状态的服务器。

3、**处理投票**。针对每一个投票，服务器都需要将别人的投票和自己的投票进行PK，PK规则如下：

* 优先检查ZXID。ZXID比较大的服务器优先作为Leader。
* 如果ZXID相同，那么就比较SID。SID较大的服务器作为Leader服务器。

对于Server1而言，它的投票是(1, 0)，接收Server2的投票为(2, 0)，首先会比较两者的ZXID，均为0，再比较SID，此时Server2的SID最大，于是更新自己的投票为(2, 0)，然后重新投票，对于Server2而言，其无须更新自己的投票，只是再次向集群中所有机器发出上一次投票信息即可。

4、**统计投票**。每次投票后，服务器都会统计投票信息，判断是否已经有过半机器接受到相同的投票信息，对于Server1、Server2而言，都统计出集群中已经有两台机器接受了(2, 0)的投票信息，此时便认为已经选出了Leader。

5、**改变服务器状态**。一旦确定了Leader，每个服务器就会更新自己的状态，如果是Follower，那么就变更为FOLLOWING，如果是Leader，就变更为LEADING。

# 五、服务器运行时期的Leader选举

在Zookeeper运行期间，Leader与非Leader服务器各司其职，即便当有非Leader服务器宕机或新加入，此时也不会影响Leader，但是一旦Leader服务器挂了，那么整个集群将暂停对外服务，进入新一轮Leader选举，其过程和启动时期的Leader选举过程基本一致。

假设正在运行的有Server1、Server2、Server3三台服务器，当前Leader是Server2，若某一时刻Leader挂了，此时便开始Leader选举。

选举过程如下

1、**变更状态**。Leader挂后，余下的非Observer服务器都会将自己的服务器状态变更为LOOKING，然后开始进入Leader选举过程。

2、**每个Server会发出一个投票**。在运行期间，每个服务器上的ZXID可能不同，此时假定Server1的ZXID为123，Server3的ZXID为122；在第一轮投票中，Server1和Server3都会投自己，产生投票(1, 123)，(3, 122)，然后各自将投票发送给集群中所有机器。

3、**接收来自各个服务器的投票**。与启动时过程相同。

4、**处理投票**。与启动时过程相同，此时，Server1将会成为Leader。

5、**统计投票**。与启动时过程相同。

6、**改变服务器的状态**。与启动时过程相同。

# 六、算法的核心内容

**1、案例分析**

假定Zookeeper由5台机器组成，SID分别为1、2、3、4、5，ZXID分别为9、9、9、8、8，并且此时SID为2的机器是Leader机器，某一时刻，1、2所在机器出现故障，因此集群开始进行Leader选举。

在第一次投票时，由于还无法检测到集群中其他机器的状态信息，因此每台机器都会将自己作为被推举的对象进行投票，于是SID为3、4、5的机器投票情况分别为(3, 9)，(4, 8)， (5, 8)。

**2、变更投票**

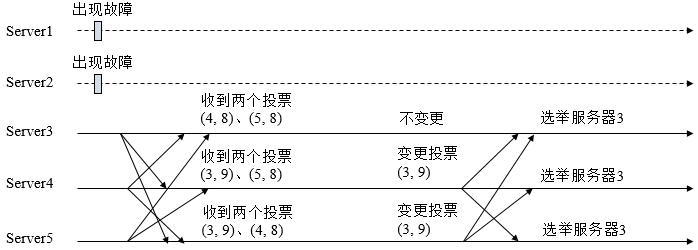
集群中每台机器发出投票后，也会收到急群众其他机器的投票，每台机器会根据一定规则来处理收到的其他机器的投票，并以此来决定是否需要变更自己的投票，这个规则也是整个Leader选举算法的核心所在，其中术语描述如下：

* **vote\_sid**：接收到的投票中**所推举**Leader服务器的SID。
* **vote\_zxid**：接收到的投票中**所推举**Leader服务器的ZXID。
* **self\_sid**：**当前**服务器自己的SID。
* **self\_zxid**：**当前**服务器自己的ZXID。

每次对收到的投票的处理，都是对(vote\_sid, vote\_zxid)和(self\_sid, self\_zxid)对比的过程。

* **规则1**：如果vote\_zxid > self\_zxid，就认可当前收到的投票，并再次将该投票发送出去。
* **规则2**：如果vote\_zxid < self\_zxid，那么坚持自己的投票，不做任何变更。
* **规则3**：如果vote\_zxid = self\_zxid，那么就对比两者的SID，如果vote\_sid > self\_sid，那么就认可当前收到的投票，并再次将该投票发送出去。
* **规则4**：如果vote\_zxid = self\_zxid，并且vote\_sid < self\_sid，那么坚持自己的投票，不做任何变更。

结合上面规则，给出下面的集群变更过程。



**3、确定Leader**

经过第二轮投票后，集群中的每台机器都会再次接收到其他机器的投票，然后开始统计投票，如果一台机器收到了**超过半数的相同投票**，那么这个投票对应的SID机器即为Leader。此时Server3将成为Leader。

**4、总结**

由上面规则可知，**通常那台服务器上的数据越新（ZXID会越大），其成为Leader的可能性越大，也就越能够保证数据的恢复**。如果ZXID相同，则SID越大机会越大。

# 七、扩展问题

## 1、为什么在Zookeeper中Server 数目一般为奇数？

我们知道在Zookeeper中 Leader 选举算法需要超过半数的同意。也就是多数 Server 写成功，则任务数据写成功。

①如果有3个Server，则最多允许1个Server 挂掉。

②如果有4个Server，则同样最多允许1个Server挂掉。

既然3个或者4个Server，同样最多允许1个Server挂掉，那么它们的可靠性是一样的，所以选择奇数个Server即可，这里选择3个Server。