ZooKeeper是一个分布式的，开放源码的分布式应用程序协调服务，它包含一个简单的原语集，分布式应用程序可以基于它实现同步服务，配置维护 和命名服务等。Zookeeper是hadoop的一个子项目，其发展历程无需赘述。在分布式应用中，由于工程师不能很好地使用锁机制，以及基于消息的协 调机制不适合在某些应用中使用，因此需要有一种可靠的、可扩展的、分布式的、可配置的协调机制来统一系统的状态。Zookeeper的目的就在于此。本文 简单分析zookeeper的工作原理，对于如何使用zookeeper不是本文讨论的重点。

**1 Zookeeper的基本概念**

**1.1 角色**

Zookeeper中的角色主要有以下三类，如下表所示：

[](http://stblog.baidu-tech.com/wp-content/uploads/wp-display-data.php?filename=71321447452.jpg&type=image%2Fjpeg&width=573&height=257)

系统模型如图所示：

[](http://stblog.baidu-tech.com/wp-content/uploads/wp-display-data.php?filename=11321444551.jpg&type=image%2Fjpeg&width=559&height=275)

**1.2 设计目的**

1.最终一致性：client不论连接到哪个Server，展示给它都是同一个视图，这是zookeeper最重要的性能。

2 .可靠性：具有简单、健壮、良好的性能，如果消息m被到一台服务器接受，那么它将被所有的服务器接受。

3 .实时性：Zookeeper保证客户端将在一个时间间隔范围内获得服务器的更新信息，或者服务器失效的信息。但由于网络延时等原因，Zookeeper不能保证两个客户端能同时得到刚更新的数据，如果需要最新数据，应该在读数据之前调用sync()接口。

4 .等待无关（wait-free）：慢的或者失效的client不得干预快速的client的请求，使得每个client都能有效的等待。

5.原子性：更新只能成功或者失败，没有中间状态。

6 .顺序性：包括全局有序和偏序两种：全局有序是指如果在一台服务器上消息a在消息b前发布，则在所有Server上消息a都将在消息b前被发布；偏序是指如果一个消息b在消息a后被同一个发送者发布，a必将排在b前面。

**2 ZooKeeper的工作原理**

Zookeeper的核心是原子广播，这个机制保证了各个Server之间的同步。实现这个机制的协议叫做Zab协议。Zab协议有两种模式，它们 分别是恢复模式（选主）和广播模式（同步）。当服务启动或者在领导者崩溃后，Zab就进入了恢复模式，当领导者被选举出来，且大多数Server完成了和 leader的状态同步以后，恢复模式就结束了。状态同步保证了leader和Server具有相同的系统状态。

为了保证事务的顺序一致性，zookeeper采用了递增的事务id号（zxid）来标识事务。所有的提议（proposal）都在被提出的时候加 上了zxid。实现中zxid是一个64位的数字，它高32位是epoch用来标识leader关系是否改变，每次一个leader被选出来，它都会有一 个新的epoch，标识当前属于那个leader的统治时期。低32位用于递增计数。

每个Server在工作过程中有三种状态：

* LOOKING：当前Server不知道leader是谁，正在搜寻
* LEADING：当前Server即为选举出来的leader
* FOLLOWING：leader已经选举出来，当前Server与之同步

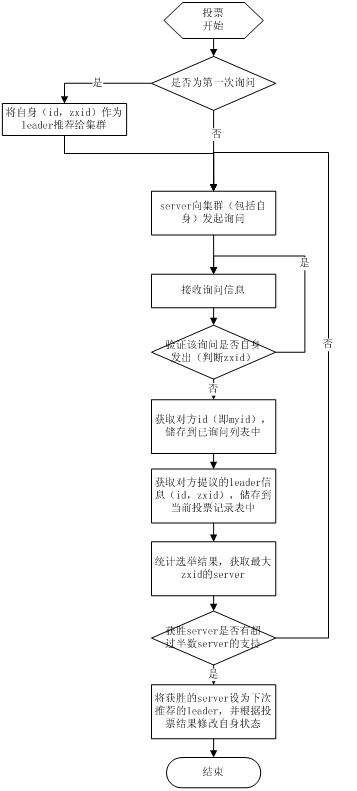
**2.1 选主流程**

当leader崩溃或者leader失去大多数的follower，这时候zk进入恢复模式，恢复模式需要重新选举出一个新的leader，让所有 的Server都恢复到一个正确的状态。Zk的选举算法有两种：一种是基于basic paxos实现的，另外一种是基于fast paxos算法实现的。系统默认的选举算法为fast paxos。先介绍basic paxos流程：

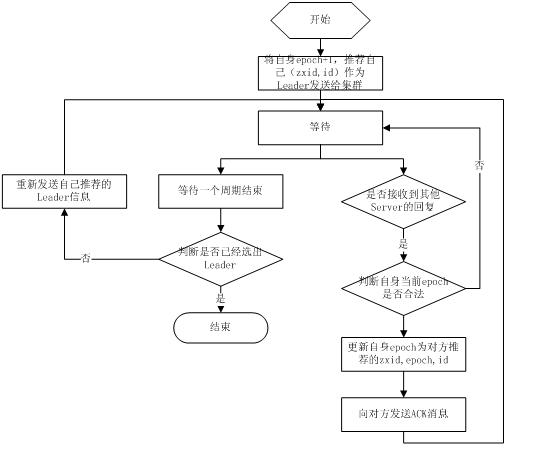
1. 1 .选举线程由当前Server发起选举的线程担任，其主要功能是对投票结果进行统计，并选出推荐的Server；
2. 2 .选举线程首先向所有Server发起一次询问(包括自己)；
3. 3 .选举线程收到回复后，验证是否是自己发起的询问(验证zxid是否一致)，然后获取对方的id(myid)，并存储到当前询问对象列表中，最后获取对方提议的leader相关信息(id,zxid)，并将这些信息存储到当次选举的投票记录表中；
4. 4.  收到所有Server回复以后，就计算出zxid最大的那个Server，并将这个Server相关信息设置成下一次要投票的Server；
5. 5.  线程将当前zxid最大的Server设置为当前Server要推荐的Leader，如果此时获胜的Server获得n/2 + 1的Server票数， 设置当前推荐的leader为获胜的Server，将根据获胜的Server相关信息设置自己的状态，否则，继续这个过程，直到leader被选举出来。

通过流程分析我们可以得出：要使Leader获得多数Server的支持，则Server总数必须是奇数2n+1，且存活的Server的数目不得少于n+1.

每个Server启动后都会重复以上流程。在恢复模式下，如果是刚从崩溃状态恢复的或者刚启动的server还会从磁盘快照中恢复数据和会话信息，zk会记录事务日志并定期进行快照，方便在恢复时进行状态恢复。选主的具体流程图如下所示：

[](http://stblog.baidu-tech.com/wp-content/uploads/wp-display-data.php?filename=21321444769.jpg&type=image%2Fjpeg&width=357&height=791)

fast paxos流程是在选举过程中，某Server首先向所有Server提议自己要成为leader，当其它Server收到提议以后，解决epoch和 zxid的冲突，并接受对方的提议，然后向对方发送接受提议完成的消息，重复这个流程，最后一定能选举出Leader。其流程图如下所示：

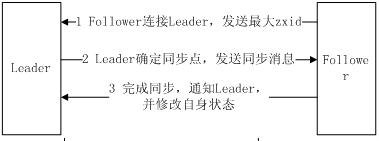
[](http://stblog.baidu-tech.com/wp-content/uploads/wp-display-data.php?filename=31321444830.jpg&type=image%2Fjpeg&width=533&height=451)

**2.2 同步流程**

选完leader以后，zk就进入状态同步过程。

1. 1. leader等待server连接；
2. 2 .Follower连接leader，将最大的zxid发送给leader；
3. 3 .Leader根据follower的zxid确定同步点；
4. 4 .完成同步后通知follower 已经成为uptodate状态；
5. 5 .Follower收到uptodate消息后，又可以重新接受client的请求进行服务了。

流程图如下所示：

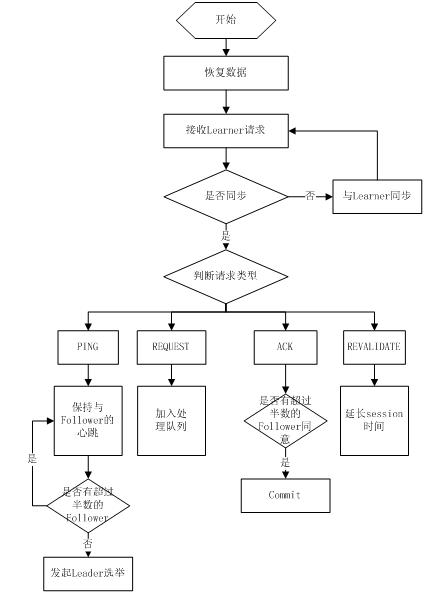
[](http://stblog.baidu-tech.com/wp-content/uploads/wp-display-data.php?filename=41321446291.jpg&type=image%2Fjpeg&width=379&height=141)

**2.3 工作流程**

**2.3.1 Leader工作流程**

Leader主要有三个功能：

1. 1 .恢复数据；
2. 2 .维持与Learner的心跳，接收Learner请求并判断Learner的请求消息类型；
3. 3 .Learner的消息类型主要有PING消息、REQUEST消息、ACK消息、REVALIDATE消息，根据不同的消息类型，进行不同的处理。

PING消息是指Learner的心跳信息；REQUEST消息是Follower发送的提议信息，包括写请求及同步请求；ACK消息是 Follower的对提议的回复，超过半数的Follower通过，则commit该提议；REVALIDATE消息是用来延长SESSION有效时间。  
Leader的工作流程简图如下所示，在实际实现中，流程要比下图复杂得多，启动了三个线程来实现功能。[](http://stblog.baidu-tech.com/wp-content/uploads/wp-display-data.php?filename=51321446379.jpg&type=image%2Fjpeg&width=425&height=594)

**2.3.2 Follower工作流程**

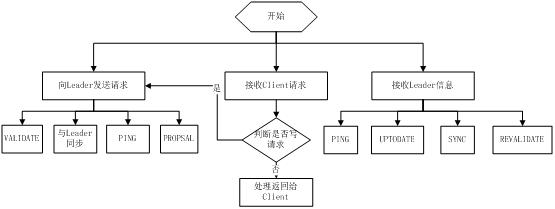
Follower主要有四个功能：

1. 1. 向Leader发送请求（PING消息、REQUEST消息、ACK消息、REVALIDATE消息）；
2. 2 .接收Leader消息并进行处理；
3. 3 .接收Client的请求，如果为写请求，发送给Leader进行投票；
4. 4 .返回Client结果。

Follower的消息循环处理如下几种来自Leader的消息：

1. **1 .PING**消息： 心跳消息；
2. **2 .PROPOSAL**消息：Leader发起的提案，要求Follower投票；
3. **3 .COMMIT**消息：服务器端最新一次提案的信息；
4. **4 .UPTODATE**消息：表明同步完成；
5. **5 .REVALIDATE**消息：根据Leader的REVALIDATE结果，关闭待revalidate的session还是允许其接受消息；
6. **6 .SYNC**消息：返回SYNC结果到客户端，这个消息最初由客户端发起，用来强制得到最新的更新。

Follower的工作流程简图如下所示，在实际实现中，Follower是通过5个线程来实现功能的。

[](http://stblog.baidu-tech.com/wp-content/uploads/wp-display-data.php?filename=61321446523.jpg&type=image%2Fjpeg&width=555&height=209)

对于observer的流程不再叙述，observer流程和Follower的唯一不同的地方就是observer不会参加leader发起的投票。