[lanqiu5ge](https://www.cnblogs.com/lanqiu5ge/)

* [博客园](https://www.cnblogs.com/)
* [首页](https://www.cnblogs.com/lanqiu5ge/)
* [新随笔](https://i.cnblogs.com/EditPosts.aspx?opt=1)
* [联系](https://msg.cnblogs.com/send/lanqiu5ge)
* [订阅](https://www.cnblogs.com/lanqiu5ge/rss/)
* [管理](https://i.cnblogs.com/)

随笔 - 75  文章 - 0  评论 - 2

[**Zookeeper面试题**](https://www.cnblogs.com/lanqiu5ge/p/9405601.html)

**目录**

* [1. ZooKeeper是什么？](https://www.cnblogs.com/lanqiu5ge/p/9405601.html#_label0)
* [2. ZooKeeper提供了什么？](https://www.cnblogs.com/lanqiu5ge/p/9405601.html#_label1)
* [3. Zookeeper文件系统](https://www.cnblogs.com/lanqiu5ge/p/9405601.html#_label2)
* [4. ZAB协议？](https://www.cnblogs.com/lanqiu5ge/p/9405601.html#_label3)
* [5. 四种类型的数据节点 Znode](https://www.cnblogs.com/lanqiu5ge/p/9405601.html#_label4)
* [6. Zookeeper Watcher 机制 -- 数据变更通知](https://www.cnblogs.com/lanqiu5ge/p/9405601.html#_label5)
* [7. 客户端注册Watcher实现](https://www.cnblogs.com/lanqiu5ge/p/9405601.html#_label6)
* [8. 服务端处理Watcher实现](https://www.cnblogs.com/lanqiu5ge/p/9405601.html#_label7)
* [9. 客户端回调Watcher](https://www.cnblogs.com/lanqiu5ge/p/9405601.html#_label8)
* [10. ACL权限控制机制](https://www.cnblogs.com/lanqiu5ge/p/9405601.html#_label9)
  + [UGO（User/Group/Others）](https://www.cnblogs.com/lanqiu5ge/p/9405601.html#_label9_0)
  + [ACL（Access Control List）访问控制列表](https://www.cnblogs.com/lanqiu5ge/p/9405601.html#_label9_1)
* [11. Chroot特性](https://www.cnblogs.com/lanqiu5ge/p/9405601.html#_label10)
* [12. 会话管理](https://www.cnblogs.com/lanqiu5ge/p/9405601.html#_label11)
* [13. 服务器角色](https://www.cnblogs.com/lanqiu5ge/p/9405601.html#_label12)
  + [Leader](https://www.cnblogs.com/lanqiu5ge/p/9405601.html#_label12_0)
  + [Follower](https://www.cnblogs.com/lanqiu5ge/p/9405601.html#_label12_1)
  + [Observer](https://www.cnblogs.com/lanqiu5ge/p/9405601.html#_label12_2)
* [14. Zookeeper 下 Server工作状态](https://www.cnblogs.com/lanqiu5ge/p/9405601.html#_label13)
* [15. Leader 选举](https://www.cnblogs.com/lanqiu5ge/p/9405601.html#_label14)
* [16. 数据同步](https://www.cnblogs.com/lanqiu5ge/p/9405601.html#_label15)
  + [直接差异化同步（DIFF同步）](https://www.cnblogs.com/lanqiu5ge/p/9405601.html#_label15_0)
  + [先回滚再差异化同步（TRUNC+DIFF同步）](https://www.cnblogs.com/lanqiu5ge/p/9405601.html#_label15_1)
  + [仅回滚同步（TRUNC同步）](https://www.cnblogs.com/lanqiu5ge/p/9405601.html#_label15_2)
  + [全量同步（SNAP同步）](https://www.cnblogs.com/lanqiu5ge/p/9405601.html#_label15_3)
* [17. zookeeper是如何保证事务的顺序一致性的？](https://www.cnblogs.com/lanqiu5ge/p/9405601.html#_label16)
* [18. 分布式集群中为什么会有Master？](https://www.cnblogs.com/lanqiu5ge/p/9405601.html#_label17)
* [19. zk节点宕机如何处理？](https://www.cnblogs.com/lanqiu5ge/p/9405601.html#_label18)
* [20. zookeeper负载均衡和nginx负载均衡区别](https://www.cnblogs.com/lanqiu5ge/p/9405601.html#_label19)
* [21. Zookeeper有哪几种几种部署模式？](https://www.cnblogs.com/lanqiu5ge/p/9405601.html#_label20)
* [22. 集群最少要几台机器，集群规则是怎样的?](https://www.cnblogs.com/lanqiu5ge/p/9405601.html#_label21)
* [23. 集群支持动态添加机器吗？](https://www.cnblogs.com/lanqiu5ge/p/9405601.html#_label22)
* [24. Zookeeper对节点的watch监听通知是永久的吗？为什么不是永久的?](https://www.cnblogs.com/lanqiu5ge/p/9405601.html#_label23)
* [25. Zookeeper的java客户端都有哪些？](https://www.cnblogs.com/lanqiu5ge/p/9405601.html#_label24)
* [26. chubby是什么，和zookeeper比你怎么看？](https://www.cnblogs.com/lanqiu5ge/p/9405601.html#_label25)
* [27. 说几个zookeeper常用的命令。](https://www.cnblogs.com/lanqiu5ge/p/9405601.html#_label26)
* [28. ZAB和Paxos算法的联系与区别？](https://www.cnblogs.com/lanqiu5ge/p/9405601.html#_label27)
* [29. Zookeeper的典型应用场景](https://www.cnblogs.com/lanqiu5ge/p/9405601.html#_label28)
  + [1. 数据发布/订阅](https://www.cnblogs.com/lanqiu5ge/p/9405601.html#_label28_0)
  + [2. 负载均衡](https://www.cnblogs.com/lanqiu5ge/p/9405601.html#_label28_1)

[回到顶部](https://www.cnblogs.com/lanqiu5ge/p/9405601.html#_labelTop)

**1. ZooKeeper是什么？**

ZooKeeper是一个开放源码的分布式协调服务，它是集群的管理者，监视着集群中各个节点的状态根据节点提交的反馈进行下一步合理操作。最终，将简单易用的接口和性能高效、功能稳定的系统提供给用户。

分布式应用程序可以基于Zookeeper实现诸如数据发布/订阅、负载均衡、命名服务、分布式协调/通知、集群管理、Master选举、分布式锁和分布式队列等功能。

Zookeeper保证了如下分布式一致性特性：

* 顺序一致性
* 原子性
* 单一视图
* 可靠性
* 实时性（最终一致性）

客户端的读请求可以被集群中的任意一台机器处理，**如果读请求在节点上注册了监听器，这个监听器也是由所连接的zookeeper机器来处理**。对于写请求，这些请求会同时发给其他zookeeper机器并且达成一致后，请求才会返回成功。因此，随着zookeeper的集群机器增多，读请求的吞吐会提高但是写请求的吞吐会下降。

有序性是zookeeper中非常重要的一个特性，所有的更新都是全局有序的，每个更新都有一个唯一的时间戳，这个时间戳称为zxid（Zookeeper Transaction Id）。而读请求只会相对于更新有序，也就是读请求的返回结果中会带有这个zookeeper最新的zxid。

[回到顶部](https://www.cnblogs.com/lanqiu5ge/p/9405601.html#_labelTop)

**2. ZooKeeper提供了什么？**

1、文件系统  
2、通知机制

[回到顶部](https://www.cnblogs.com/lanqiu5ge/p/9405601.html#_labelTop)

**3. Zookeeper文件系统**

Zookeeper提供一个多层级的节点命名空间（节点称为znode）。与文件系统不同的是，这些节点都可以设置关联的数据，而文件系统中只有文件节点可以存放数据而目录节点不行。  
Zookeeper为了保证高吞吐和低延迟，在内存中维护了这个树状的目录结构，**这种特性使得Zookeeper不能用于存放大量的数据，每个节点的存放数据上限为1M**。

[回到顶部](https://www.cnblogs.com/lanqiu5ge/p/9405601.html#_labelTop)

**4. ZAB协议？**

ZAB协议是为分布式协调服务Zookeeper专门设计的一种支持崩溃恢复的原子广播协议。

ZAB协议包括两种基本的模式：崩溃恢复和消息广播。

当整个zookeeper集群刚刚启动或者Leader服务器宕机、重启或者网络故障导致不存在过半的服务器与Leader服务器保持正常通信时，所有进程（服务器）进入崩溃恢复模式，首先选举产生新的Leader服务器，然后集群中Follower服务器开始与新的Leader服务器进行数据同步，当集群中超过半数机器与该Leader服务器完成数据同步之后，退出恢复模式进入消息广播模式，Leader服务器开始接收客户端的事务请求生成事物提案来进行事务请求处理。

[回到顶部](https://www.cnblogs.com/lanqiu5ge/p/9405601.html#_labelTop)

**5. 四种类型的数据节点 Znode**

* PERSISTENT-持久节点  
  除非手动删除，否则节点一直存在于Zookeeper上
* EPHEMERAL-临时节点  
  临时节点的生命周期与客户端会话绑定，一旦客户端会话失效（客户端与zookeeper连接断开不一定会话失效），那么这个客户端创建的所有临时节点都会被移除。
* PERSISTENT\_SEQUENTIAL-持久顺序节点  
  基本特性同持久节点，只是增加了顺序属性，节点名后边会追加一个由父节点维护的自增整型数字。
* EPHEMERAL\_SEQUENTIAL-临时顺序节点  
  基本特性同临时节点，增加了顺序属性，节点名后边会追加一个由父节点维护的自增整型数字。

[回到顶部](https://www.cnblogs.com/lanqiu5ge/p/9405601.html#_labelTop)

**6. Zookeeper Watcher 机制 -- 数据变更通知**

Zookeeper允许客户端向服务端的某个Znode注册一个Watcher监听，当服务端的一些指定事件触发了这个Watcher，服务端会向指定客户端发送一个事件通知来实现分布式的通知功能，然后客户端根据Watcher通知状态和事件类型做出业务上的改变。

工作机制：

* 客户端注册watcher
* 服务端处理watcher
* 客户端回调watcher

Watcher特性总结：

1. 一次性  
   无论是服务端还是客户端，一旦一个Watcher被触发，Zookeeper都会将其从相应的存储中移除。这样的设计有效的减轻了服务端的压力，不然对于更新非常频繁的节点，服务端会不断的向客户端发送事件通知，无论对于网络还是服务端的压力都非常大。
2. 客户端串行执行  
   客户端Watcher回调的过程是一个串行同步的过程。
3. 轻量
   * Watcher通知非常简单，只会告诉客户端发生了事件，而不会说明事件的具体内容。
   * 客户端向服务端注册Watcher的时候，并不会把客户端真实的Watcher对象实体传递到服务端，仅仅是在客户端请求中使用boolean类型属性进行了标记。
4. watcher event异步发送watcher的通知事件从server发送到client是异步的，这就存在一个问题，不同的客户端和服务器之间通过socket进行通信，由于网络延迟或其他因素导致客户端在不通的时刻监听到事件，由于Zookeeper本身提供了ordering guarantee，即客户端监听事件后，才会感知它所监视znode发生了变化。所以我们使用Zookeeper不能期望能够监控到节点每次的变化。Zookeeper只能保证最终的一致性，而无法保证强一致性。
5. 注册watcher getData、exists、getChildren
6. 触发watcher create、delete、setData
7. 当一个客户端连接到一个新的服务器上时，watch将会被以任意会话事件触发。当与一个服务器失去连接的时候，是无法接收到watch的。而当client重新连接时，如果需要的话，所有先前注册过的watch，都会被重新注册。通常这是完全透明的。只有在一个特殊情况下，watch可能会丢失：对于一个未创建的znode的exist watch，如果在客户端断开连接期间被创建了，并且随后在客户端连接上之前又删除了，这种情况下，这个watch事件可能会被丢失。

[回到顶部](https://www.cnblogs.com/lanqiu5ge/p/9405601.html#_labelTop)

**7. 客户端注册Watcher实现**

1. 调用getData()/getChildren()/exist()三个API，传入Watcher对象
2. 标记请求request，封装Watcher到WatchRegistration
3. 封装成Packet对象，发服务端发送request
4. 收到服务端响应后，将Watcher注册到ZKWatcherManager中进行管理
5. 请求返回，完成注册。

[回到顶部](https://www.cnblogs.com/lanqiu5ge/p/9405601.html#_labelTop)

**8. 服务端处理Watcher实现**

1. 服务端接收Watcher并存储  
   接收到客户端请求，处理请求判断是否需要注册Watcher，需要的话将数据节点的节点路径和ServerCnxn（ServerCnxn代表一个客户端和服务端的连接，实现了Watcher的process接口，此时可以看成一个Watcher对象）存储在WatcherManager的WatchTable和watch2Paths中去。
2. Watcher触发  
   以服务端接收到 setData() 事务请求触发NodeDataChanged事件为例：
   * 封装WatchedEvent  
     将通知状态（SyncConnected）、事件类型（NodeDataChanged）以及节点路径封装成一个WatchedEvent对象
   * 查询Watcher  
     从WatchTable中根据节点路径查找Watcher
   * 没找到；说明没有客户端在该数据节点上注册过Watcher
   * 找到；提取并从WatchTable和Watch2Paths中删除对应Watcher（**从这里可以看出Watcher在服务端是一次性的，触发一次就失效了**）
3. 调用process方法来触发Watcher  
   这里process主要就是通过ServerCnxn对应的TCP连接发送Watcher事件通知。

[回到顶部](https://www.cnblogs.com/lanqiu5ge/p/9405601.html#_labelTop)

**9. 客户端回调Watcher**

客户端SendThread线程接收事件通知，交由EventThread线程回调Watcher。客户端的Watcher机制同样是一次性的，一旦被触发后，该Watcher就失效了。

[回到顶部](https://www.cnblogs.com/lanqiu5ge/p/9405601.html#_labelTop)

**10. ACL权限控制机制**

**UGO（User/Group/Others）**

目前在Linux/Unix文件系统中使用，也是使用最广泛的权限控制方式。是一种粗粒度的文件系统权限控制模式。

**ACL（Access Control List）访问控制列表**

包括三个方面：

* 权限模式（Scheme）
  + IP：从IP地址粒度进行权限控制
  + Digest：最常用，用类似于 username:password 的权限标识来进行权限配置，便于区分不同应用来进行权限控制
  + World：最开放的权限控制方式，是一种特殊的digest模式，只有一个权限标识“world:anyone”
  + Super：超级用户
* 授权对象  
  授权对象指的是权限赋予的用户或一个指定实体，例如IP地址或是机器灯。
* 权限 Permission
  + CREATE：数据节点创建权限，允许授权对象在该Znode下创建子节点
  + DELETE：子节点删除权限，允许授权对象删除该数据节点的子节点
  + READ：数据节点的读取权限，允许授权对象访问该数据节点并读取其数据内容或子节点列表等
  + WRITE：数据节点更新权限，允许授权对象对该数据节点进行更新操作
  + ADMIN：数据节点管理权限，允许授权对象对该数据节点进行ACL相关设置操作

[回到顶部](https://www.cnblogs.com/lanqiu5ge/p/9405601.html#_labelTop)

**11. Chroot特性**

3.2.0版本后，添加了 Chroot特性，该特性允许每个客户端为自己设置一个命名空间。如果一个客户端设置了Chroot，那么该客户端对服务器的任何操作，都将会被限制在其自己的命名空间下。

通过设置Chroot，能够将一个客户端应用于Zookeeper服务端的一颗子树相对应，在那些多个应用公用一个Zookeeper进群的场景下，对实现不同应用间的相互隔离非常有帮助。

[回到顶部](https://www.cnblogs.com/lanqiu5ge/p/9405601.html#_labelTop)

**12. 会话管理**

分桶策略：将类似的会话放在同一区块中进行管理，以便于Zookeeper对会话进行不同区块的隔离处理以及同一区块的统一处理。

分配原则：每个会话的“下次超时时间点”（ExpirationTime）

计算公式：

ExpirationTime\_ = currentTime + sessionTimeout

ExpirationTime = (ExpirationTime\_ / ExpirationInrerval + 1) \* ExpirationInterval , ExpirationInterval 是指 Zookeeper 会话超时检查时间间隔，默认 tickTime

[回到顶部](https://www.cnblogs.com/lanqiu5ge/p/9405601.html#_labelTop)

**13. 服务器角色**

**Leader**

* 事务请求的唯一调度和处理者，保证集群事务处理的顺序性
* 集群内部各服务的调度者

**Follower**

* 处理客户端的非事务请求，转发事务请求给Leader服务器
* 参与事务请求Proposal的投票
* 参与Leader选举投票

**Observer**

3.3.0版本以后引入的一个服务器角色，在不影响集群事务处理能力的基础上提升集群的非事务处理能力

* 处理客户端的非事务请求，转发事务请求给Leader服务器
* 不参与任何形式的投票

[回到顶部](https://www.cnblogs.com/lanqiu5ge/p/9405601.html#_labelTop)

**14. Zookeeper 下 Server工作状态**

服务器具有四种状态，分别是LOOKING、FOLLOWING、LEADING、OBSERVING。

* 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，因此需要进入Leader选举状态。
* FOLLOWING：跟随者状态。表明当前服务器角色是Follower。
* LEADING：领导者状态。表明当前服务器角色是Leader。
* OBSERVING：观察者状态。表明当前服务器角色是Observer。

[回到顶部](https://www.cnblogs.com/lanqiu5ge/p/9405601.html#_labelTop)

**15. Leader 选举**

Leader选举是保证分布式数据一致性的关键所在。当Zookeeper集群中的一台服务器出现以下两种情况之一时，需要进入Leader选举。

　　(1) 服务器初始化启动。

　　(2) 服务器运行期间无法和Leader保持连接。

　　下面就两种情况进行分析讲解。

　　1. 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myid和ZXID，使用(myid, 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myid。myid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myid，此时Server2的myid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。

　　2. 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服务器都会讲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。

　　2.2 Leader选举算法分析

　　在3.4.0后的Zookeeper的版本只保留了TCP版本的FastLeaderElection选举算法。当一台机器进入Leader选举时，当前集群可能会处于以下两种状态

　　　　· 集群中已经存在Leader。

　　　　· 集群中不存在Leader。

　　对于集群中已经存在Leader而言，此种情况一般都是某台机器启动得较晚，在其启动之前，集群已经在正常工作，对这种情况，该机器试图去选举Leader时，会被告知当前服务器的Leader信息，对于该机器而言，仅仅需要和Leader机器建立起连接，并进行状态同步即可。而在集群中不存在Leader情况下则会相对复杂，其步骤如下

　　(1) 第一次投票。无论哪种导致进行Leader选举，集群的所有机器都处于试图选举出一个Leader的状态，即LOOKING状态，LOOKING机器会向所有其他机器发送消息，该消息称为投票。投票中包含了SID（服务器的唯一标识）和ZXID（事务ID），(SID, ZXID)形式来标识一次投票信息。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，每台机器都会将自己作为投票对象，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。每台机器发出投票后，也会收到其他机器的投票，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。

　　　　规则一：如果vote\_zxid大于self\_zxid，就认可当前收到的投票，并再次将该投票发送出去。

　　　　规则二：如果vote\_zxid小于self\_zxid，那么坚持自己的投票，不做任何变更。

　　　　规则三：如果vote\_zxid等于self\_zxid，那么就对比两者的SID，如果vote\_sid大于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。

　　　　规则四：如果vote\_zxid等于self\_zxid，并且vote\_sid小于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。

　　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。

　　2.3 Leader选举实现细节

　　1. 服务器状态

　　服务器具有四种状态，分别是LOOKING、FOLLOWING、LEADING、OBSERVING。

　　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，因此需要进入Leader选举状态。

　　FOLLOWING：跟随者状态。表明当前服务器角色是Follower。

　　LEADING：领导者状态。表明当前服务器角色是Leader。

　　OBSERVING：观察者状态。表明当前服务器角色是Observer。

　　2. 投票数据结构

　　每个投票中包含了两个最基本的信息，所推举服务器的SID和ZXID，投票（Vote）在Zookeeper中包含字段如下

　　id：被推举的Leader的SID。

　　zxid：被推举的Leader事务ID。

　　electionEpoch：逻辑时钟，用来判断多个投票是否在同一轮选举周期中，该值在服务端是一个自增序列，每次进入新一轮的投票后，都会对该值进行加1操作。

　　peerEpoch：被推举的Leader的epoch。

　　state：当前服务器的状态。

　　3. QuorumCnxManager：网络I/O

　　每台服务器在启动的过程中，会启动一个QuorumPeerManager，负责各台服务器之间的底层Leader选举过程中的网络通信。

　　(1) 消息队列。QuorumCnxManager内部维护了一系列的队列，用来保存接收到的、待发送的消息以及消息的发送器，除接收队列以外，其他队列都按照SID分组形成队列集合，如一个集群中除了自身还有3台机器，那么就会为这3台机器分别创建一个发送队列，互不干扰。

　　　　· recvQueue：消息接收队列，用于存放那些从其他服务器接收到的消息。

　　　　· queueSendMap：消息发送队列，用于保存那些待发送的消息，按照SID进行分组。

　　　　· senderWorkerMap：发送器集合，每个SenderWorker消息发送器，都对应一台远程Zookeeper服务器，负责消息的发送，也按照SID进行分组。

　　　　· lastMessageSent：最近发送过的消息，为每个SID保留最近发送过的一个消息。

　　(2) 建立连接。为了能够相互投票，Zookeeper集群中的所有机器都需要两两建立起网络连接。QuorumCnxManager在启动时会创建一个ServerSocket来监听Leader选举的通信端口(默认为3888)。开启监听后，Zookeeper能够不断地接收到来自其他服务器的创建连接请求，在接收到其他服务器的TCP连接请求时，会进行处理。为了避免两台机器之间重复地创建TCP连接，Zookeeper只允许SID大的服务器主动和其他机器建立连接，否则断开连接。在接收到创建连接请求后，服务器通过对比自己和远程服务器的SID值来判断是否接收连接请求，如果当前服务器发现自己的SID更大，那么会断开当前连接，然后自己主动和远程服务器建立连接。一旦连接建立，就会根据远程服务器的SID来创建相应的消息发送器SendWorker和消息接收器RecvWorker，并启动。

　　(3) 消息接收与发送。消息接收：由消息接收器RecvWorker负责，由于Zookeeper为每个远程服务器都分配一个单独的RecvWorker，因此，每个RecvWorker只需要不断地从这个TCP连接中读取消息，并将其保存到recvQueue队列中。消息发送：由于Zookeeper为每个远程服务器都分配一个单独的SendWorker，因此，每个SendWorker只需要不断地从对应的消息发送队列中获取出一个消息发送即可，同时将这个消息放入lastMessageSent中。在SendWorker中，一旦Zookeeper发现针对当前服务器的消息发送队列为空，那么此时需要从lastMessageSent中取出一个最近发送过的消息来进行再次发送，这是为了解决接收方在消息接收前或者接收到消息后服务器挂了，导致消息尚未被正确处理。同时，Zookeeper能够保证接收方在处理消息时，会对重复消息进行正确的处理。

　　4. FastLeaderElection：选举算法核心

　　· 外部投票：特指其他服务器发来的投票。

　　· 内部投票：服务器自身当前的投票。

　　· 选举轮次：Zookeeper服务器Leader选举的轮次，即logicalclock。

　　· PK：对内部投票和外部投票进行对比来确定是否需要变更内部投票。

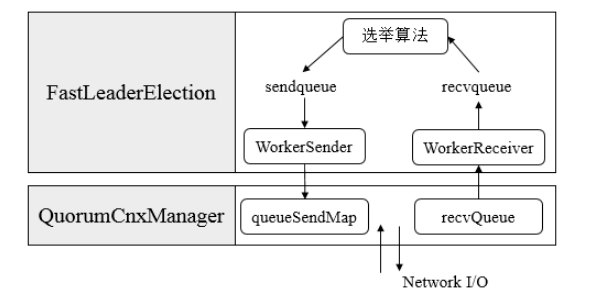
　　(1) 选票管理

　　· sendqueue：选票发送队列，用于保存待发送的选票。

　　· recvqueue：选票接收队列，用于保存接收到的外部投票。

　　· WorkerReceiver：选票接收器。其会不断地从QuorumCnxManager中获取其他服务器发来的选举消息，并将其转换成一个选票，然后保存到recvqueue中，在选票接收过程中，如果发现该外部选票的选举轮次小于当前服务器的，那么忽略该外部投票，同时立即发送自己的内部投票。

　　· WorkerSender：选票发送器，不断地从sendqueue中获取待发送的选票，并将其传递到底层QuorumCnxManager中。

　　(2) 算法核心  


　　上图展示了FastLeaderElection模块是如何与底层网络I/O进行交互的。Leader选举的基本流程如下

　　1. 自增选举轮次。Zookeeper规定所有有效的投票都必须在同一轮次中，在开始新一轮投票时，会首先对logicalclock进行自增操作。

　　2. 初始化选票。在开始进行新一轮投票之前，每个服务器都会初始化自身的选票，并且在初始化阶段，每台服务器都会将自己推举为Leader。

　　3. 发送初始化选票。完成选票的初始化后，服务器就会发起第一次投票。Zookeeper会将刚刚初始化好的选票放入sendqueue中，由发送器WorkerSender负责发送出去。

　　4. 接收外部投票。每台服务器会不断地从recvqueue队列中获取外部选票。如果服务器发现无法获取到任何外部投票，那么就会立即确认自己是否和集群中其他服务器保持着有效的连接，如果没有连接，则马上建立连接，如果已经建立了连接，则再次发送自己当前的内部投票。

　　5. 判断选举轮次。在发送完初始化选票之后，接着开始处理外部投票。在处理外部投票时，会根据选举轮次来进行不同的处理。

　　　　· 外部投票的选举轮次大于内部投票。若服务器自身的选举轮次落后于该外部投票对应服务器的选举轮次，那么就会立即更新自己的选举轮次(logicalclock)，并且清空所有已经收到的投票，然后使用初始化的投票来进行PK以确定是否变更内部投票。最终再将内部投票发送出去。

　　　　· 外部投票的选举轮次小于内部投票。若服务器接收的外选票的选举轮次落后于自身的选举轮次，那么Zookeeper就会直接忽略该外部投票，不做任何处理，并返回步骤4。

　　　　· 外部投票的选举轮次等于内部投票。此时可以开始进行选票PK。

　　6. 选票PK。在进行选票PK时，符合任意一个条件就需要变更投票。

　　　　· 若外部投票中推举的Leader服务器的选举轮次大于内部投票，那么需要变更投票。

　　　　· 若选举轮次一致，那么就对比两者的ZXID，若外部投票的ZXID大，那么需要变更投票。

　　　　· 若两者的ZXID一致，那么就对比两者的SID，若外部投票的SID大，那么就需要变更投票。

　　7. 变更投票。经过PK后，若确定了外部投票优于内部投票，那么就变更投票，即使用外部投票的选票信息来覆盖内部投票，变更完成后，再次将这个变更后的内部投票发送出去。

　　8. 选票归档。无论是否变更了投票，都会将刚刚收到的那份外部投票放入选票集合recvset中进行归档。recvset用于记录当前服务器在本轮次的Leader选举中收到的所有外部投票（按照服务队的SID区别，如{(1, vote1), (2, vote2)...}）。

　　9. 统计投票。完成选票归档后，就可以开始统计投票，统计投票是为了统计集群中是否已经有过半的服务器认可了当前的内部投票，如果确定已经有过半服务器认可了该投票，则终止投票。否则返回步骤4。

　　10. 更新服务器状态。若已经确定可以终止投票，那么就开始更新服务器状态，服务器首选判断当前被过半服务器认可的投票所对应的Leader服务器是否是自己，若是自己，则将自己的服务器状态更新为LEADING，若不是，则根据具体情况来确定自己是FOLLOWING或是OBSERVING。

　　以上10个步骤就是FastLeaderElection的核心，其中步骤4-9会经过几轮循环，直到有Leader选举产生。

[回到顶部](https://www.cnblogs.com/lanqiu5ge/p/9405601.html#_labelTop)

**16. 数据同步**

整个集群完成Leader选举之后，Learner（Follower和Observer的统称）回向Leader服务器进行注册。当Learner服务器想Leader服务器完成注册后，进入数据同步环节。

数据同步流程：（均以消息传递的方式进行）

i. Learner向Learder注册

ii. 数据同步

iii. 同步确认

**Zookeeper的数据同步通常分为四类**：

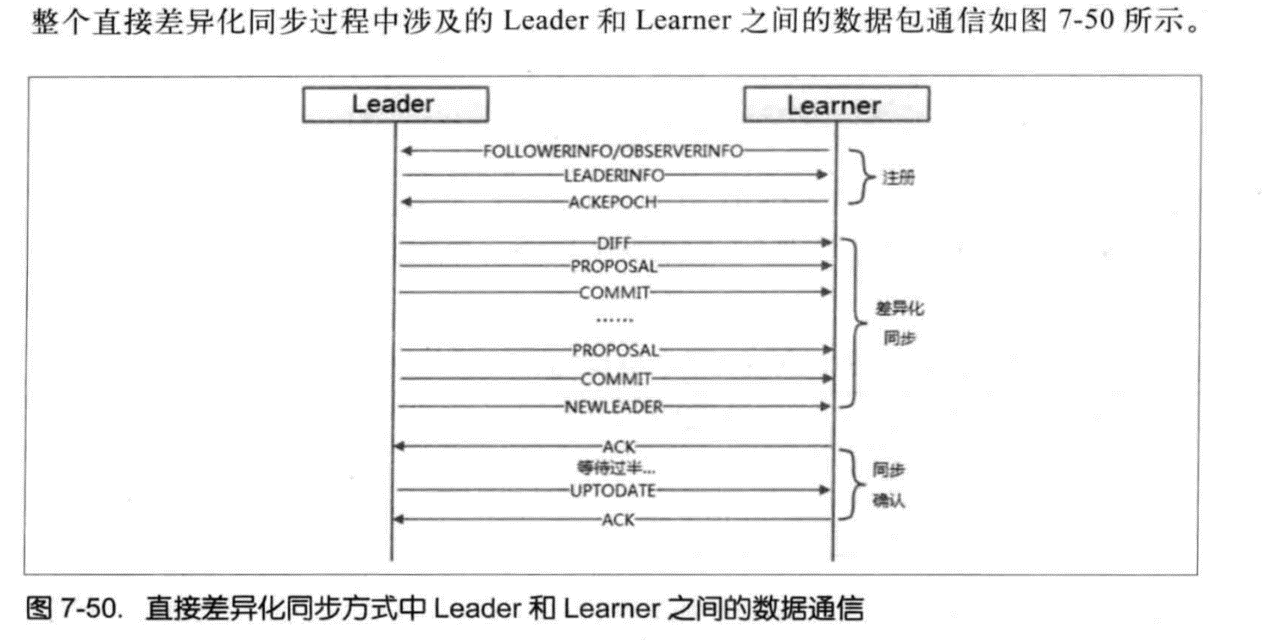
* 直接差异化同步（DIFF同步）
* 先回滚再差异化同步（TRUNC+DIFF同步）
* 仅回滚同步（TRUNC同步）
* 全量同步（SNAP同步）

在进行数据同步前，Leader服务器会完成数据同步初始化：

* peerLastZxid：从learner服务器注册时发送的ACKEPOCH消息中提取lastZxid（该Learner服务器最后处理的ZXID）
* minCommittedLog：Leader服务器Proposal缓存队列committedLog中最小ZXID
* maxCommittedLog：Leader服务器Proposal缓存队列committedLog中最大ZXID

**直接差异化同步（DIFF同步）**

**场景**：peerLastZxid介于minCommittedLog和maxCommittedLog之间



**先回滚再差异化同步（TRUNC+DIFF同步）**

**场景**：当新的Leader服务器发现某个Learner服务器包含了一条自己没有的事务记录，那么就需要让该Learner服务器进行事务回滚--回滚到Leader服务器上存在的，同时也是最接近于peerLastZxid的ZXID

**仅回滚同步（TRUNC同步）**

**场景**：peerLastZxid 大于 maxCommittedLog

**全量同步（SNAP同步）**

**场景一**：peerLastZxid 小于 minCommittedLog  
**场景二**：Leader服务器上没有Proposal缓存队列且peerLastZxid不等于lastProcessZxid

[回到顶部](https://www.cnblogs.com/lanqiu5ge/p/9405601.html#_labelTop)

**17. zookeeper是如何保证事务的顺序一致性的？**

zookeeper采用了全局递增的事务Id来标识，所有的proposal（提议）都在被提出的时候加上了zxid，zxid实际上是一个64位的数字，高32位是epoch（时期; 纪元; 世; 新时代）用来标识leader周期，如果有新的leader产生出来，epoch会自增，低32位用来递增计数。当新产生proposal的时候，会依据数据库的两阶段过程，首先会向其他的server发出事务执行请求，如果超过半数的机器都能执行并且能够成功，那么就会开始执行。

[回到顶部](https://www.cnblogs.com/lanqiu5ge/p/9405601.html#_labelTop)

**18. 分布式集群中为什么会有Master？**

在分布式环境中，有些业务逻辑只需要集群中的某一台机器进行执行，其他的机器可以共享这个结果，这样可以大大减少重复计算，提高性能，于是就需要进行leader选举。

[回到顶部](https://www.cnblogs.com/lanqiu5ge/p/9405601.html#_labelTop)

**19. zk节点宕机如何处理？**

Zookeeper本身也是集群，推荐配置不少于3个服务器。Zookeeper自身也要保证当一个节点宕机时，其他节点会继续提供服务。  
如果是一个Follower宕机，还有2台服务器提供访问，因为Zookeeper上的数据是有多个副本的，数据并不会丢失；  
如果是一个Leader宕机，Zookeeper会选举出新的Leader。  
ZK集群的机制是只要超过半数的节点正常，集群就能正常提供服务。只有在ZK节点挂得太多，只剩一半或不到一半节点能工作，集群才失效。  
所以  
3个节点的cluster可以挂掉1个节点(leader可以得到2票>1.5)  
2个节点的cluster就不能挂掉任何1个节点了(leader可以得到1票<=1)

[回到顶部](https://www.cnblogs.com/lanqiu5ge/p/9405601.html#_labelTop)

**20. zookeeper负载均衡和nginx负载均衡区别**

zk的负载均衡是可以调控，nginx只是能调权重，其他需要可控的都需要自己写插件；但是nginx的吞吐量比zk大很多，应该说按业务选择用哪种方式。

[回到顶部](https://www.cnblogs.com/lanqiu5ge/p/9405601.html#_labelTop)

**21. Zookeeper有哪几种几种部署模式？**

部署模式：单机模式、伪集群模式、集群模式。

[回到顶部](https://www.cnblogs.com/lanqiu5ge/p/9405601.html#_labelTop)

**22. 集群最少要几台机器，集群规则是怎样的?**

集群规则为2N+1台，N>0，即3台。

[回到顶部](https://www.cnblogs.com/lanqiu5ge/p/9405601.html#_labelTop)

**23. 集群支持动态添加机器吗？**

其实就是水平扩容了，Zookeeper在这方面不太好。两种方式：

* 全部重启：关闭所有Zookeeper服务，修改配置之后启动。不影响之前客户端的会话。
* 逐个重启：在过半存活即可用的原则下，一台机器重启不影响整个集群对外提供服务。这是比较常用的方式。

3.5版本开始支持动态扩容。

[回到顶部](https://www.cnblogs.com/lanqiu5ge/p/9405601.html#_labelTop)

**24. Zookeeper对节点的watch监听通知是永久的吗？为什么不是永久的?**

不是。官方声明：一个Watch事件是一个一次性的触发器，当被设置了Watch的数据发生了改变的时候，则服务器将这个改变发送给设置了Watch的客户端，以便通知它们。

为什么不是永久的，举个例子，如果服务端变动频繁，而监听的客户端很多情况下，每次变动都要通知到所有的客户端，给网络和服务器造成很大压力。  
一般是客户端执行getData(“/节点A”,true)，如果节点A发生了变更或删除，客户端会得到它的watch事件，但是在之后节点A又发生了变更，而客户端又没有设置watch事件，就不再给客户端发送。  
在实际应用中，很多情况下，我们的客户端不需要知道服务端的每一次变动，我只要最新的数据即可。

[回到顶部](https://www.cnblogs.com/lanqiu5ge/p/9405601.html#_labelTop)

**25. Zookeeper的java客户端都有哪些？**

java客户端：zk自带的zkclient及Apache开源的Curator。

[回到顶部](https://www.cnblogs.com/lanqiu5ge/p/9405601.html#_labelTop)

**26. chubby是什么，和zookeeper比你怎么看？**

chubby是google的，完全实现paxos算法，不开源。zookeeper是chubby的开源实现，使用zab协议，paxos算法的变种。

[回到顶部](https://www.cnblogs.com/lanqiu5ge/p/9405601.html#_labelTop)

**27. 说几个zookeeper常用的命令。**

常用命令：ls get set create delete等。

[回到顶部](https://www.cnblogs.com/lanqiu5ge/p/9405601.html#_labelTop)

**28. ZAB和Paxos算法的联系与区别？**

* 相同点：
  + 两者都存在一个类似于Leader进程的角色，由其负责协调多个Follower进程的运行
  + Leader进程都会等待超过半数的Follower做出正确的反馈后，才会将一个提案进行提交
  + ZAB协议中，每个Proposal中都包含一个 epoch 值来代表当前的Leader周期，Paxos中名字为Ballot
* 不同点：  
  ZAB用来构建高可用的分布式数据主备系统（Zookeeper），Paxos是用来构建分布式一致性状态机系统。

[回到顶部](https://www.cnblogs.com/lanqiu5ge/p/9405601.html#_labelTop)

**29. Zookeeper的典型应用场景**

Zookeeper是一个典型的发布/订阅模式的分布式数据管理与协调框架，开发人员可以使用它来进行分布式数据的发布和订阅。

通过对Zookeeper中丰富的数据节点进行交叉使用，配合Watcher事件通知机制，可以非常方便的构建一系列分布式应用中年都会涉及的核心功能，如：

* 数据发布/订阅
* 负载均衡
* 命名服务
* 分布式协调/通知
* 集群管理
* Master选举
* 分布式锁
* 分布式队列

**1. 数据发布/订阅**

**介绍**

数据发布/订阅系统，即所谓的配置中心，顾名思义就是发布者发布数据供订阅者进行数据订阅。

**目的**

* 动态获取数据（配置信息）
* 实现数据（配置信息）的集中式管理和数据的动态更新

**设计模式**

* Push 模式
* Pull 模式

**数据（配置信息）特性：**

* 数据量通常比较小
* 数据内容在运行时会发生动态更新
* 集群中各机器共享，配置一致

如：机器列表信息、运行时开关配置、数据库配置信息等

**基于Zookeeper的实现方式**

1. 数据存储：将数据（配置信息）存储到Zookeeper上的一个数据节点
2. 数据获取：应用在启动初始化节点从Zookeeper数据节点读取数据，并在该节点上注册一个数据变更Watcher
3. 数据变更：当变更数据时，更新Zookeeper对应节点数据，Zookeeper会将数据变更通知发到各客户端，客户端接到通知后重新读取变更后的数据即可。

**2. 负载均衡**

zk的命名服务  
命名服务是指通过指定的名字来获取资源或者服务的地址，利用zk创建一个全局的路径，这个路径就可以作为一个名字，指向集群中的集群，提供的服务的地址，或者一个远程的对象等等。

分布式通知和协调  
对于系统调度来说：操作人员发送通知实际是通过控制台改变某个节点的状态，然后zk将这些变化发送给注册了这个节点的watcher的所有客户端。  
对于执行情况汇报：每个工作进程都在某个目录下创建一个临时节点。并携带工作的进度数据，这样汇总的进程可以监控目录子节点的变化获得工作进度的实时的全局情况。

7.zk的命名服务（文件系统）  
命名服务是指通过指定的名字来获取资源或者服务的地址，利用zk创建一个全局的路径，即是唯一的路径，这个路径就可以作为一个名字，指向集群中的集群，提供的服务的地址，或者一个远程的对象等等。

8.zk的配置管理（文件系统、通知机制）  
程序分布式的部署在不同的机器上，将程序的配置信息放在zk的znode下，当有配置发生改变时，也就是znode发生变化时，可以通过改变zk中某个目录节点的内容，利用watcher通知给各个客户端，从而更改配置。

9.Zookeeper集群管理（文件系统、通知机制）  
所谓集群管理无在乎两点：是否有机器退出和加入、选举master。  
对于第一点，所有机器约定在父目录下创建临时目录节点，然后监听父目录节点的子节点变化消息。一旦有机器挂掉，该机器与 zookeeper的连接断开，其所创建的临时目录节点被删除，所有其他机器都收到通知：某个兄弟目录被删除，于是，所有人都知道：它上船了。  
新机器加入也是类似，所有机器收到通知：新兄弟目录加入，highcount又有了，对于第二点，我们稍微改变一下，所有机器创建临时顺序编号目录节点，每次选取编号最小的机器作为master就好。

10.Zookeeper分布式锁（文件系统、通知机制）  
有了zookeeper的一致性文件系统，锁的问题变得容易。锁服务可以分为两类，一个是保持独占，另一个是控制时序。  
对于第一类，我们将zookeeper上的一个znode看作是一把锁，通过createznode的方式来实现。所有客户端都去创建 /distribute\_lock 节点，最终成功创建的那个客户端也即拥有了这把锁。用完删除掉自己创建的distribute\_lock 节点就释放出锁。  
对于第二类， /distribute\_lock 已经预先存在，所有客户端在它下面创建临时顺序编号目录节点，和选master一样，编号最小的获得锁，用完删除，依次方便。

11.获取分布式锁的流程  
clipboard.png

在获取分布式锁的时候在locker节点下创建临时顺序节点，释放锁的时候删除该临时节点。客户端调用createNode方法在locker下创建临时顺序节点，  
然后调用getChildren(“locker”)来获取locker下面的所有子节点，注意此时不用设置任何Watcher。客户端获取到所有的子节点path之后，如果发现自己创建的节点在所有创建的子节点序号最小，那么就认为该客户端获取到了锁。如果发现自己创建的节点并非locker所有子节点中最小的，说明自己还没有获取到锁，此时客户端需要找到比自己小的那个节点，然后对其调用exist()方法，同时对其注册事件监听器。之后，让这个被关注的节点删除，则客户端的Watcher会收到相应通知，此时再次判断自己创建的节点是否是locker子节点中序号最小的，如果是则获取到了锁，如果不是则重复以上步骤继续获取到比自己小的一个节点并注册监听。当前这个过程中还需要许多的逻辑判断。  
clipboard.png

代码的实现主要是基于互斥锁，获取分布式锁的重点逻辑在于BaseDistributedLock，实现了基于Zookeeper实现分布式锁的细节。

12.Zookeeper队列管理（文件系统、通知机制）  
两种类型的队列：  
1、同步队列，当一个队列的成员都聚齐时，这个队列才可用，否则一直等待所有成员到达。  
2、队列按照 FIFO 方式进行入队和出队操作。  
第一类，在约定目录下创建临时目录节点，监听节点数目是否是我们要求的数目。  
第二类，和分布式锁服务中的控制时序场景基本原理一致，入列有编号，出列按编号。在特定的目录下创建PERSISTENT\_SEQUENTIAL节点，创建成功时Watcher通知等待的队列，队列删除序列号最小的节点用以消费。此场景下Zookeeper的znode用于消息存储，znode存储的数据就是消息队列中的消息内容，SEQUENTIAL序列号就是消息的编号，按序取出即可。由于创建的节点是持久化的，所以不必担心队列消息的丢失问题。

分类: [面试题整理](https://www.cnblogs.com/lanqiu5ge/category/1261777.html), [Zookeeper](https://www.cnblogs.com/lanqiu5ge/category/1284498.html)

标签: [Zookeeper](https://www.cnblogs.com/lanqiu5ge/tag/Zookeeper/), [面试题](https://www.cnblogs.com/lanqiu5ge/tag/%E9%9D%A2%E8%AF%95%E9%A2%98/), [分布式](https://www.cnblogs.com/lanqiu5ge/tag/%E5%88%86%E5%B8%83%E5%BC%8F/)

[**好文要顶**](javascript:void(0);) [**关注我**](javascript:void(0);) [**收藏该文**](javascript:void(0);) **[](javascript:void(0);)** **[](javascript:void(0);)**

[](https://home.cnblogs.com/u/lanqiu5ge/)

[lanqiu5ge](https://home.cnblogs.com/u/lanqiu5ge/)  
[关注 - 7](https://home.cnblogs.com/u/lanqiu5ge/followees/)  
[粉丝 - 8](https://home.cnblogs.com/u/lanqiu5ge/followers/)

[+加关注](javascript:void(0);)

8

0

[«](https://www.cnblogs.com/lanqiu5ge/p/9472197.html)上一篇： [理解volatile与synchronized](https://www.cnblogs.com/lanqiu5ge/p/9472197.html)   
[»](https://www.cnblogs.com/lanqiu5ge/p/9442454.html)下一篇： [（转）浅谈架构-从传统走向分布式](https://www.cnblogs.com/lanqiu5ge/p/9442454.html)

posted @ 2018-08-02 10:26 [lanqiu5ge](https://www.cnblogs.com/lanqiu5ge/) 阅读(19568) 评论(0) [编辑](https://i.cnblogs.com/EditPosts.aspx?postid=9405601) [收藏](javascript:void(0))

[刷新评论](javascript:void(0);)[刷新页面](https://www.cnblogs.com/lanqiu5ge/p/9405601.html)[返回顶部](https://www.cnblogs.com/lanqiu5ge/p/9405601.html#top)

（评论功能已被禁用）

[【推荐】超50万C++/C#源码: 大型实时仿真组态图形源码](http://www.ucancode.com/index.htm)  
[【推荐】零基础轻松玩转云上产品，获壕礼加返百元大礼](http://clickc.admaster.com.cn/c/a131574,b3595115,c1705,i0,m101,8a1,8b3,h)  
[【推荐】华为IoT平台开发者套餐9.9元起，购买即送免费课程](http://clickc.admaster.com.cn/c/a131575,b3595121,c1705,i0,m101,8a1,8b3,h)

**相关博文：**  
· [ZooKeeper的简单理解](https://www.cnblogs.com/simple-focus/p/6575233.html)  
· [zookeeper一二三](https://www.cnblogs.com/checkout/p/10559660.html)  
· [zookeeper-01概述](https://www.cnblogs.com/zhanglianghhh/p/9094252.html)  
· [zookeeper简单小节](https://www.cnblogs.com/spaceapp/p/10322458.html)  
· [初始zookeeper与集群搭建实例](https://www.cnblogs.com/dennisit/p/4141342.html)

**最新 IT 新闻**:   
· [华为方舟编译器今日开源：OpenSource网站已上线](https://news.cnblogs.com/n/630410/)   
· [人类与“永动机”的千年恩怨（二）：存在“额外”的能量吗？](https://news.cnblogs.com/n/630409/)   
· [通用汽车、Lyft和Waymo希望允许自动驾驶汽车不装方向盘](https://news.cnblogs.com/n/630408/)   
· [工厂将断电停产！订单已骤减至零！华映大裁员：2100人仅留50人？](https://news.cnblogs.com/n/630407/)   
· [雅加达30年后或消失？海平面上升将“重构”世界版图](https://news.cnblogs.com/n/630406/)   
» [更多新闻...](https://news.cnblogs.com/)

**公告**

昵称： [lanqiu5ge](https://home.cnblogs.com/u/lanqiu5ge/)  
园龄： [5年3个月](https://home.cnblogs.com/u/lanqiu5ge/)  
粉丝： [8](https://home.cnblogs.com/u/lanqiu5ge/followers/)  
关注： [7](https://home.cnblogs.com/u/lanqiu5ge/followees/)

[+加关注](javascript:void(0))

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | [<](javascript:void(0);) | 2019年8月 | [>](javascript:void(0);) | | | | | | | |
| **日** | **一** | **二** | **三** | **四** | **五** | **六** |
| 28 | 29 | 30 | 31 | 1 | 2 | 3 |
| 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| 18 | 19 | 20 | 21 | 22 | 23 | 24 |
| 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |

**搜索**





**常用链接**

* [我的随笔](https://www.cnblogs.com/lanqiu5ge/p/)
* [我的评论](https://www.cnblogs.com/lanqiu5ge/MyComments.html)
* [我的参与](https://www.cnblogs.com/lanqiu5ge/OtherPosts.html)
* [最新评论](https://www.cnblogs.com/lanqiu5ge/RecentComments.html)
* [我的标签](https://www.cnblogs.com/lanqiu5ge/tag/)

**我的标签**

* [Java](https://www.cnblogs.com/lanqiu5ge/tag/Java/)(9)
* [Effective C++](https://www.cnblogs.com/lanqiu5ge/tag/Effective%20C%2B%2B/)(6)
* [Redis](https://www.cnblogs.com/lanqiu5ge/tag/Redis/)(6)
* [Spring](https://www.cnblogs.com/lanqiu5ge/tag/Spring/)(5)
* [面试题](https://www.cnblogs.com/lanqiu5ge/tag/%E9%9D%A2%E8%AF%95%E9%A2%98/)(5)
* [分布式](https://www.cnblogs.com/lanqiu5ge/tag/%E5%88%86%E5%B8%83%E5%BC%8F/)(4)
* [key-value](https://www.cnblogs.com/lanqiu5ge/tag/key-value/)(4)
* [Linux](https://www.cnblogs.com/lanqiu5ge/tag/Linux/)(3)
* [C/C++](https://www.cnblogs.com/lanqiu5ge/tag/C%2FC%2B%2B/)(3)
* [JVM](https://www.cnblogs.com/lanqiu5ge/tag/JVM/)(3)
* [更多](https://www.cnblogs.com/lanqiu5ge/tag/)

**随笔分类**

* [C# .Net (3)](https://www.cnblogs.com/lanqiu5ge/category/1277367.html)
* [C/C++ (17)](https://www.cnblogs.com/lanqiu5ge/category/1277361.html)
* [Golang](https://www.cnblogs.com/lanqiu5ge/category/1300895.html)
* [Java(6)](https://www.cnblogs.com/lanqiu5ge/category/1274977.html)
* [Java多线程 (2)](https://www.cnblogs.com/lanqiu5ge/category/1277376.html)
* [Java反射(1)](https://www.cnblogs.com/lanqiu5ge/category/1293938.html)
* [JVM(1)](https://www.cnblogs.com/lanqiu5ge/category/1278365.html)
* [LeetCode题解 (2)](https://www.cnblogs.com/lanqiu5ge/category/1277373.html)
* [Linux(3)](https://www.cnblogs.com/lanqiu5ge/category/1280797.html)
* [Mysql(2)](https://www.cnblogs.com/lanqiu5ge/category/1280796.html)
* [Python (7)](https://www.cnblogs.com/lanqiu5ge/category/1277363.html)
* [Redis(5)](https://www.cnblogs.com/lanqiu5ge/category/1273512.html)
* [Spring(3)](https://www.cnblogs.com/lanqiu5ge/category/1278088.html)
* [Spring源码分析](https://www.cnblogs.com/lanqiu5ge/category/1285797.html)
* [UML(2)](https://www.cnblogs.com/lanqiu5ge/category/1281731.html)
* [Zookeeper(1)](https://www.cnblogs.com/lanqiu5ge/category/1284498.html)
* [安全](https://www.cnblogs.com/lanqiu5ge/category/1285297.html)
* [分布式架构(1)](https://www.cnblogs.com/lanqiu5ge/category/1273492.html)
* [计算机网络(4)](https://www.cnblogs.com/lanqiu5ge/category/1274981.html)
* [面经](https://www.cnblogs.com/lanqiu5ge/category/1273509.html)
* [面试题整理(1)](https://www.cnblogs.com/lanqiu5ge/category/1261777.html)
* [区块链](https://www.cnblogs.com/lanqiu5ge/category/1285295.html)
* [设计模式(3)](https://www.cnblogs.com/lanqiu5ge/category/1281633.html)
* [算法与数据结构 (12)](https://www.cnblogs.com/lanqiu5ge/category/1277370.html)
* [网络-协议 (1)](https://www.cnblogs.com/lanqiu5ge/category/1277371.html)
* [有感(1)](https://www.cnblogs.com/lanqiu5ge/category/1277645.html)

**随笔档案**

* [2018年9月(2)](https://www.cnblogs.com/lanqiu5ge/archive/2018/09.html)
* [2018年8月(27)](https://www.cnblogs.com/lanqiu5ge/archive/2018/08.html)
* [2018年2月(1)](https://www.cnblogs.com/lanqiu5ge/archive/2018/02.html)
* [2015年9月(1)](https://www.cnblogs.com/lanqiu5ge/archive/2015/09.html)
* [2015年7月(11)](https://www.cnblogs.com/lanqiu5ge/archive/2015/07.html)
* [2015年6月(16)](https://www.cnblogs.com/lanqiu5ge/archive/2015/06.html)
* [2015年4月(3)](https://www.cnblogs.com/lanqiu5ge/archive/2015/04.html)
* [2014年9月(1)](https://www.cnblogs.com/lanqiu5ge/archive/2014/09.html)
* [2014年8月(3)](https://www.cnblogs.com/lanqiu5ge/archive/2014/08.html)
* [2014年5月(6)](https://www.cnblogs.com/lanqiu5ge/archive/2014/05.html)
* [2014年4月(2)](https://www.cnblogs.com/lanqiu5ge/archive/2014/04.html)
* [2014年3月(1)](https://www.cnblogs.com/lanqiu5ge/archive/2014/03.html)
* [2013年11月(1)](https://www.cnblogs.com/lanqiu5ge/archive/2013/11.html)

**大牛博客**

* [五月的仓颉](https://www.cnblogs.com/xrq730/)
* Java Web 方面很咱的博客，值得学习。

**最新评论**

* [1. Re:读《初入阿里-10天》有感](https://www.cnblogs.com/lanqiu5ge/p/9474073.html#4185284)
* 说得很好，加油~
* --迈步出de围城
* [2. Re:读《初入阿里-10天》有感](https://www.cnblogs.com/lanqiu5ge/p/9474073.html#4042626)
* 加油~
* --JMCui

**阅读排行榜**

* [1. Zookeeper面试题(19568)](https://www.cnblogs.com/lanqiu5ge/p/9405601.html)
* [2. Redis常见面试题(1886)](https://www.cnblogs.com/lanqiu5ge/p/9442837.html)
* [3. HTTP-Http状态码301和302的区别(1506)](https://www.cnblogs.com/lanqiu5ge/p/9457074.html)
* [4. UML类图介绍以及PlantUML使用方法(886)](https://www.cnblogs.com/lanqiu5ge/p/9502419.html)
* [5. C++指针之间的赋值与转换规则总结(828)](https://www.cnblogs.com/lanqiu5ge/p/9472249.html)

**评论排行榜**

* [1. 读《初入阿里-10天》有感(2)](https://www.cnblogs.com/lanqiu5ge/p/9474073.html)

**推荐排行榜**

* [1. Zookeeper面试题(8)](https://www.cnblogs.com/lanqiu5ge/p/9405601.html)
* [2. 读《初入阿里-10天》有感(1)](https://www.cnblogs.com/lanqiu5ge/p/9474073.html)