FelixZh

仅用于个人备份，并非全部原创！！

昵称：FelixZh
园龄：3年6个月
粉丝：77
关注：7
+加关注

< 2018年3月 >

日	一	二	三	四	五	六
25	26	27	28	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

找找看
谷歌搜索

常用链接

我的随笔
我的评论
我的参与
最新评论
我的标签
更多链接

最新随笔

1. InfluxDB基本概念和操作
2. InfluxDB部署
3. Zookeeper运维小结--CancelledKeyException
4. Zookeeper源码编译为Eclipse工程（win7下Ant编译）
5. ZooKeeper Observers解决节点过多时写性能下降问题
6. ZooKeeper日志与快照文件简单分析
7. ZooKeeper Administrator's Guide A Guide to Deployment and Administration（吃别人嚼过的馍没意思，直接看官网资料）
8. ZOOKEEPER解惑
9. 部署与管理ZooKeeper（版本有点老，3.4.3）
10. 为什么zookeeper集群中节点配置个数是奇数个？

Zookeeper的功能以及工作原理

1.ZooKeeper是什么？
ZooKeeper是一个分布式的，开放源码的分布式应用程序协调服务，是Google的Chubby一个开源的实现，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

2.ZooKeeper提供了什么？

1)文件系统

2)通知机制

3.Zookeeper文件系统

每个子目录项如 NameService 都被称作为znode，和文件系统一样，我们能够自由的增加、删除znode，在一个znode下增加、删除子znode，唯一的不同在于znode是可以存储数据的。

有四种类型的znode：

1、PERSISTENT-持久化目录节点

客户端与zookeeper断开连接后，该节点依旧存在

2、PERSISTENT_SEQUENTIAL-持久化顺序编号目录节点

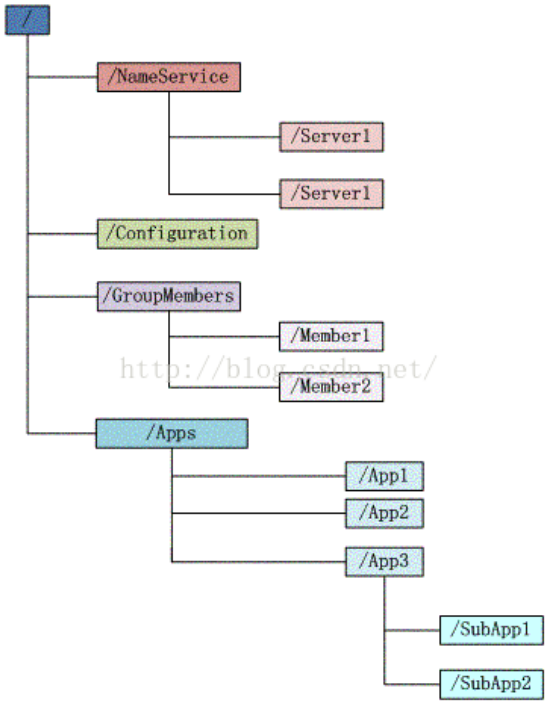
客户端与zookeeper断开连接后，该节点依旧存在，只是Zookeeper给该节点名称进行顺序编号

3、EPHEMERAL-临时目录节点

客户端与zookeeper断开连接后，该节点被删除

4、EPHEMERAL_SEQUENTIAL-临时顺序编号目录节点

客户端与zookeeper断开连接后，该节点被删除，只是Zookeeper给该节点名称进行顺序编号



<ignore_js_op>

4.Zookeeper通知机制

客户端注册监听它关心的目录节点，当目录节点发生变化（数据改变、被删除、子目录节点增加删除）时，zookeeper会通知客户端。

5.Zookeeper做了什么？

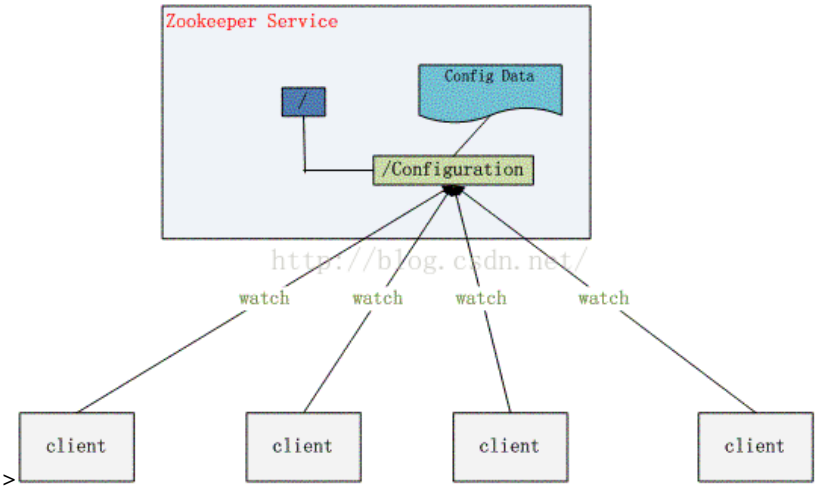
1.命名服务　2.配置管理　3.集群管理　4.分布式锁　5.队列管理

6.Zookeeper命名服务

在zookeeper的文件系统里创建一个目录，即有唯一的path。在我们使用tborg无法确定上游程序的部署机器时即可与下游程序约定好path，通过path即能互相探索发现。

7.Zookeeper的配置管理

程序总是需要配置的，如果程序分散部署在多台机器上，要逐个改变配置就变得困难。现在把这些配置全部放到zookeeper上去，保存在 Zookeeper 的某个目录节点中，然后所有相关应用程序对这个目录节点进行监听，一旦配置信息发生变化，每个应用程序就会收到 Zookeeper 的通知，然后从 Zookeeper 获取新的配置信息应用到系统中就好



<ignore_js_op>

8.Zookeeper集群管理

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GroupMembers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。

2015年10月 (2)
2015年9月 (3)
2015年8月 (19)
2015年7月 (19)
2015年6月 (5)
2015年5月 (15)
2015年4月 (10)
2015年3月 (4)
2015年2月 (3)
2015年1月 (6)
2014年12月 (6)
2014年11月 (9)
2014年10月 (1)

积分与排名

积分 - 146532
排名 - 1826

最新评论

1. Re:ContainerBase.addChild: start: org.apache.catalina.LifecycleException: Failed to start component
解决
@xuxuxu123不得不给你赞...
--FelixZh
2. Re:ContainerBase.addChild: start: org.apache.catalina.LifecycleException: Failed to start component
解决
呜呜呜拼错了，找了一天的jar包。看到这忽然想起来。。。。。。。
--xuxuxu123
3. Re:JAVA多线程实现的四种方式
@icuke可以...
--FelixZh
4. Re:JAVA多线程实现的四种方式
总结的好，可以转发么？
--icuke
5. Re:JAVA多线程实现的四种方式
实现线程的方式
1 继承thread类
2 实现runnable接口
3 实现Callable接口通过FutureTask包装器来创建Thread线程
--流星一箭

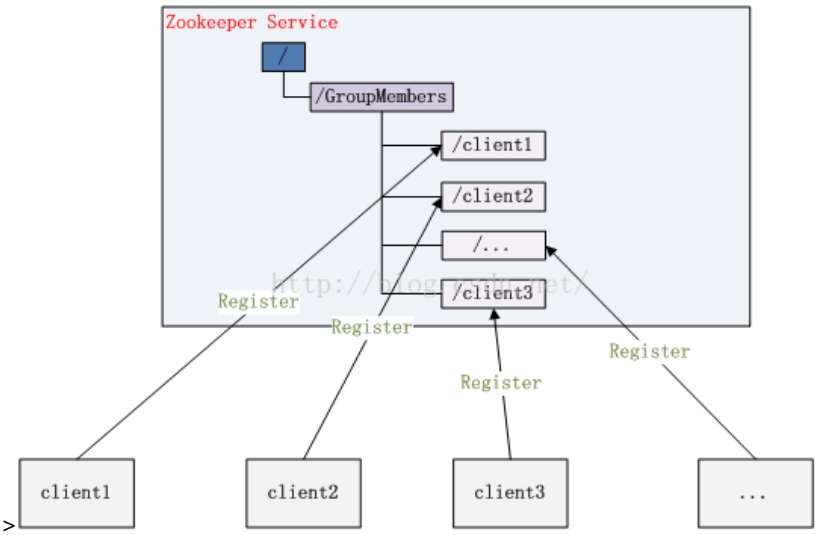
阅读排行榜

1. Zookeeper的功能以及工作原理(46360)
2. JAVA多线程实现的四种方式(40851)
3. ContainerBase.addChild: start: org.apache.catalina.LifecycleException: Failed to start component解决(31313)
4. java判断字符串是否为数字或中文或字母(13527)
5. SecureCRT上传和下载文件(下载默认目录)(8923)

评论排行榜

1. Zookeeper的功能以及工作原理(6)
2. JAVA多线程实现的四种方式(3)
3. 个人经常查阅的网站（顺序无先后，持续更新）(3)
4. ContainerBase.addChild: start: org.apache.catalina.LifecycleExce

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。



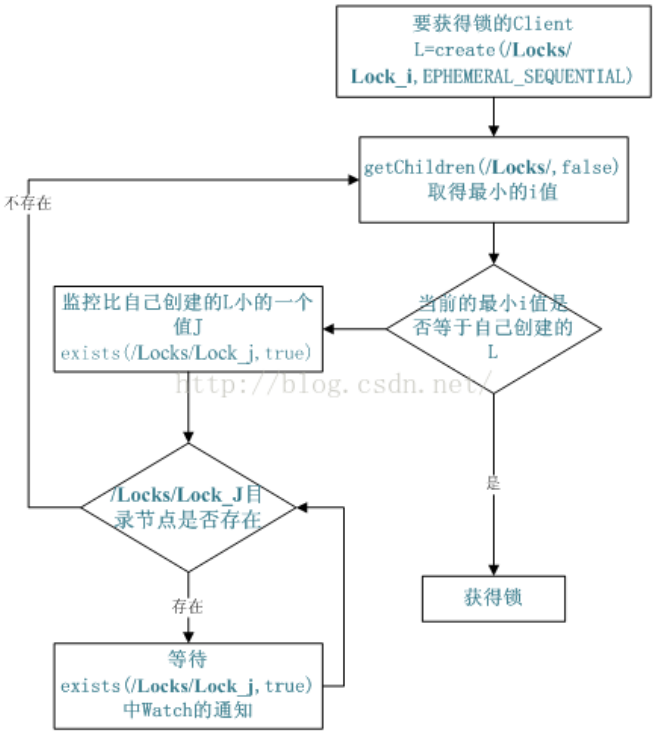
<ignore_js_op>

9.Zookeeper分布式锁

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。



<ignore_js_op>

10.Zookeeper队列管理

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。

ption: Failed to start component解决(2)
5. 深入理解Java的接口和抽象类(2)

推荐排行榜

1. JAVA多线程实现的四种方式(6)
2. Zookeeper的功能以及工作原理(3)
3. 深入理解Java的接口和抽象类(2)
4. SecureCRT上传和下载文件(下载默认目录)(1)
5. 监控Spark应用方法简介(1)

11.分布式与数据复制

Zookeeper作为一个集群提供一致的数据服务，自然，它要在所有机器间做数据复制。数据复制的好处：

1、容错：一个节点出错，不致于让整个系统停止工作，别的节点可以接管它的工作；

2、提高系统的扩展能力 ：把负载分布到多个节点上，或者增加节点来提高系统的负载能力；

3、提高性能：让客户端本地访问就近的节点，提高用户访问速度。

从客户端读写访问的透明度来看，数据复制集群系统分下面两种：

1、写主(WriteMaster) ：对数据的修改提交给指定的节点。读无此限制，可以读取任何一个节点。这种情况下客户端需要对读与写进行区别，俗称读写分离；

2、写任意(Write Any)：对数据的修改可提交给任意的节点，跟读一样。这种情况下，客户端对集群节点的角色与变化透明。

对zookeeper来说，它采用的方式是写任意。通过增加机器，它的读吞吐能力和响应能力扩展性非常好，而写，随着机器的增多吞吐能力肯定下降（这也是它建立observer的原因），而响应能力则取决于具体实现方式，是延迟复制保持最终一致性，还是立即复制快速响应。

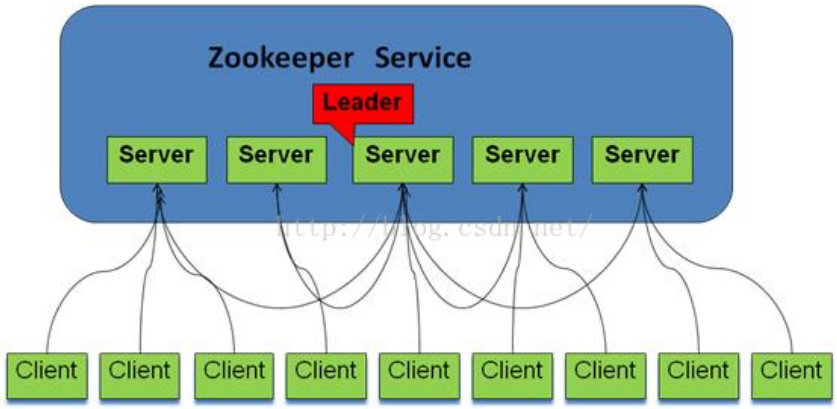
12.Zookeeper角色描述

<ignore_js_op>

角色		描述
领导者（Leader）		领导者负责进行投票的发起和决议，更新系统状态
学习者（Learner）	跟随者（Follower）	Follower 用于接收客户请求并向客户端返回结果，在选主过程中参与投票
	观察者（ObServer）	ObServer 可以接收客户端连接，将写请求转发给 leader 节点。但 ObServer 不参加投票过程，只同步 leader 的状态。ObServer 的目的是为了扩展系统，提高读取速度
客户端（Client）		请求发起方

13.Zookeeper与客户端

<ignore_js_op>



14.Zookeeper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。

15.Zookeeper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。

16.Zookeeper 下 Server工作状态

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

17.Zookeeper选主流程(basic paxos)

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。

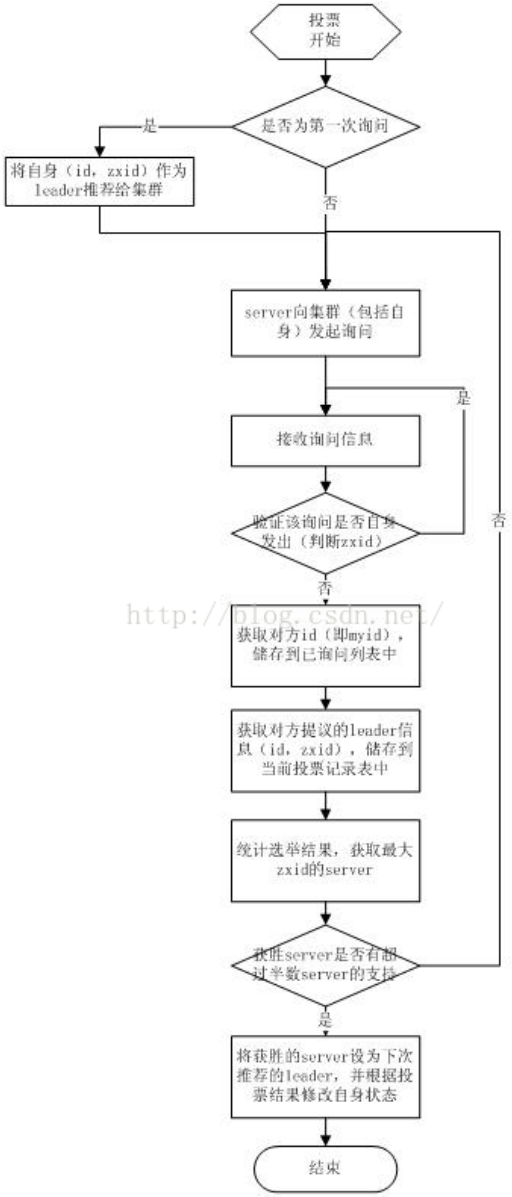
1.选举线程由当前Server发起选举的线程担任，其主要功能是对投票结果进行统计，并选出推荐的Server；

2.选举线程首先向所有Server发起一次询问(包括自己)；

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.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；

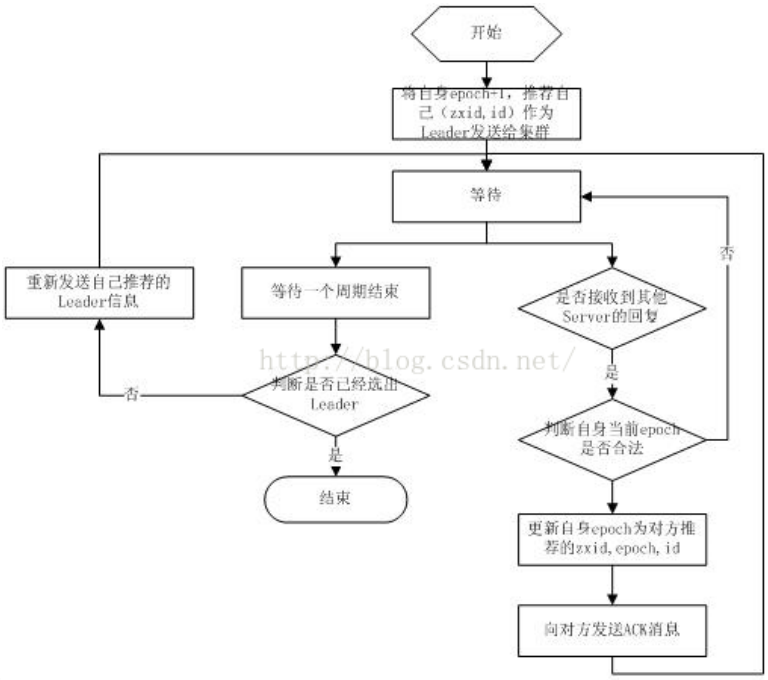
5.线程将当前zxid最大的Server设置为当前Server要推荐的Leader，如果此时获胜的Server获得n/2 + 1的Server票数，设置当前推荐的leader为获胜的Server，将根据获胜的Server相关信息设置自己的状态，否则，继续这个过程，直到leader被选举出来。 通过流程分析我们可以得出：要使Leader获得多数Server的支持，则Server总数必须是奇数2n+1，且存活的Server的数目不得少于n+1. 每个Server启动后都会重复以上流程。在恢复模式下，如果是刚从崩溃状态恢复的或者刚启动的server还会从磁盘快照中恢复数据和会话信息，zk会记录事务日志并定期进行快照，方便在恢复时进行状态恢复。选主的具体流程图所示：



<ignore_js_op>

18.Zookeeper选主流程（fast paxos）

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。



<ignore_js_op>

19.Zookeeper同步流程

选完Leader以后，zk就进入状态同步过程。

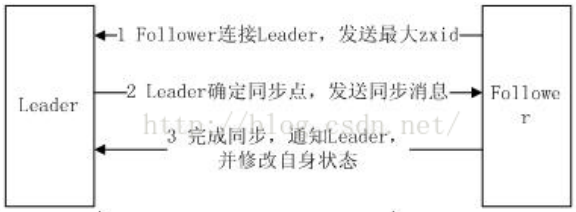
1. Leader等待server连接；

2 .Follower连接leader，将最大的zxid发送给leader；

3 .Leader根据follower的zxid确定同步点；

4 .完成同步后通知follower 已经成为uptodate状态；

5 .Follower收到uptodate消息后，又可以重新接受client的请求进行服务了。



<ignore_js_op>

20.Zookeeper工作流程-Leader

1 .恢复数据；

2 .维持与Learner的心跳，接收Learner请求并判断Learner的请求消息类型；

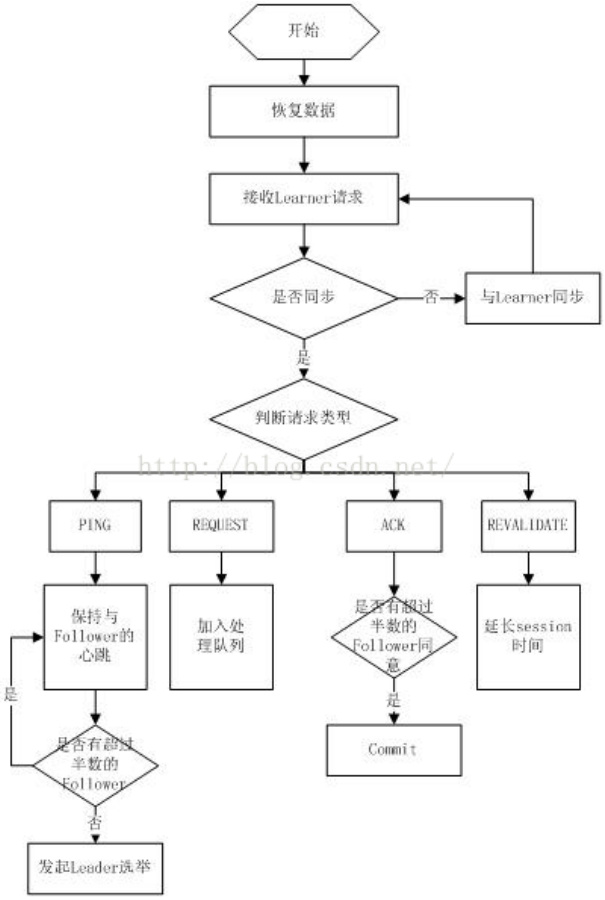
3 .Learner的消息类型主要有PING消息、REQUEST消息、ACK消息、REVALIDATE消息，根据不同的消息类型，进行不同的处理。

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。



<ignore_js_op>

21.Zookeeper工作流程-Follower

Follower主要有四个功能：

1.向Leader发送请求（PING消息、REQUEST消息、ACK消息、REVALIDATE消息）；

2.接收Leader消息并进行处理；

3.接收Client的请求，如果为写请求，发送给Leader进行投票；

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 .PING消息： 心跳消息；

2 .PROPOSAL消息：Leader发起的提案，要求Follower投票；

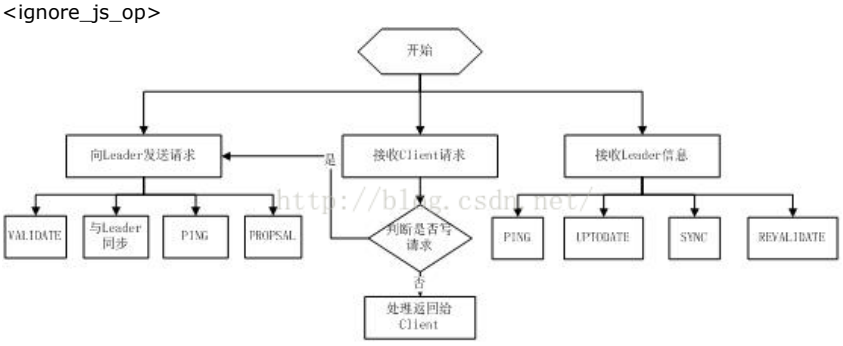
3 .COMMIT消息：服务器端最新一次提案的信息；

4 .UPTODATE消息：表明同步完成；

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 .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。

<ignore_js_op>



好了，以上就是我对zookeeper的 理解了，以后我还会继续为大家更新新的技术请大家期待吧！！！

原文链接

软件-注重思想、逻辑

分类: zookeeper

FelixZh
关注 - 7
粉丝 - 77
+加关注

3 0

« 上一篇：zookeeper
» 下一篇：zookeeper理论

posted @ 2016-09-13 17:32 FelixZh 阅读(46373) 评论(6) 编辑 收藏

评论

#1楼 2017-03-16 14:29 | 逆流的致爷

你也是刚开始学习zookeeper吗？

支持(0) 反对(0)

#2楼[楼主] 2017-03-20 08:31 | FelixZh

@ 逆流的致爷
差不多吧。只是拿来用，没做太过深入的研究

支持(0) 反对(0)

#3楼 2017-07-12 10:39 | Way_Out

看你的博客分类，感觉你的涉猎面很广啊！

支持(0) 反对(0)

#4楼[楼主] 2017-08-16 13:04 | FelixZh

@ Way_Out
用到的，就简单整理备注一下

支持(0) 反对(0)

#5楼 2017-10-13 14:32 | 大象无形01

关注了版主，没事会常来你这里看看的

支持(0) 反对(0)

#6楼 2017-10-20 23:20 | final变量

Zookeeper的功能以及工作原理- FelixZh - 博客园，写的不错不错，收藏了。

推荐下，分布式队列中间件 RocketMQ 源码解析 14 篇：http://www.yunai.me/categories/RocketMQ/?cnblog&601

锟

支持(0) 反对(0)

刷新评论 刷新页面 返回顶部

注册用户登录后才能发表评论，请 登录 或 注册，访问网站首页。