# Apache Zookeeper

作者：whoami

# zookeeper | zk是什么？

- 源代码开源（免费）
- 大型分布式系统的可靠协调系统
- 功能包括：配置维护、名字服务、分布式同步、组服务等
- **Zookeeper**是**Hadoop**生态系统中的基础组件

# zookeeper | 特点

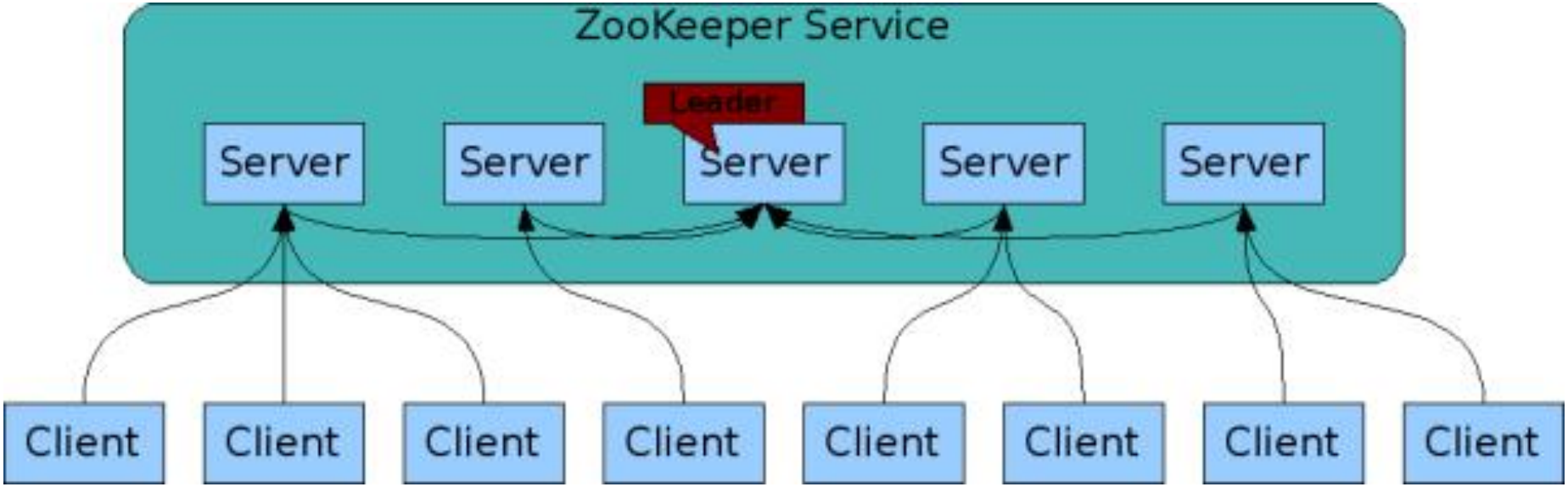
- 一致性：无论客户端连接到哪一个服务器，客户端将看到相同的 ZooKeeper 视图。
- 可靠性：一旦一个更新操作被应用，那么在客户端再次更新它之前，它的值将不会改变。
- 实时性：Zookeeper只保证最终一致性, 但是实时的一致性可以由客户端调用自己来保证,通过调用sync()方法.
- 等待无关（**wait-free**）：慢的或者失效的**client**不干预快速的**client**的请求。
- 原子性：更新操作要么成功要么失败，没有第三种结果。
- 顺序一致性：客户端的更新顺序与它们被发送的顺序相一致。

# zookeeper | 那些系统用到zk

- HDFS
- Yarn
- Storm
- Hbase
- Flume
- Kafka
- Spark
- Hive
- Drill

....

# zookeeper | 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

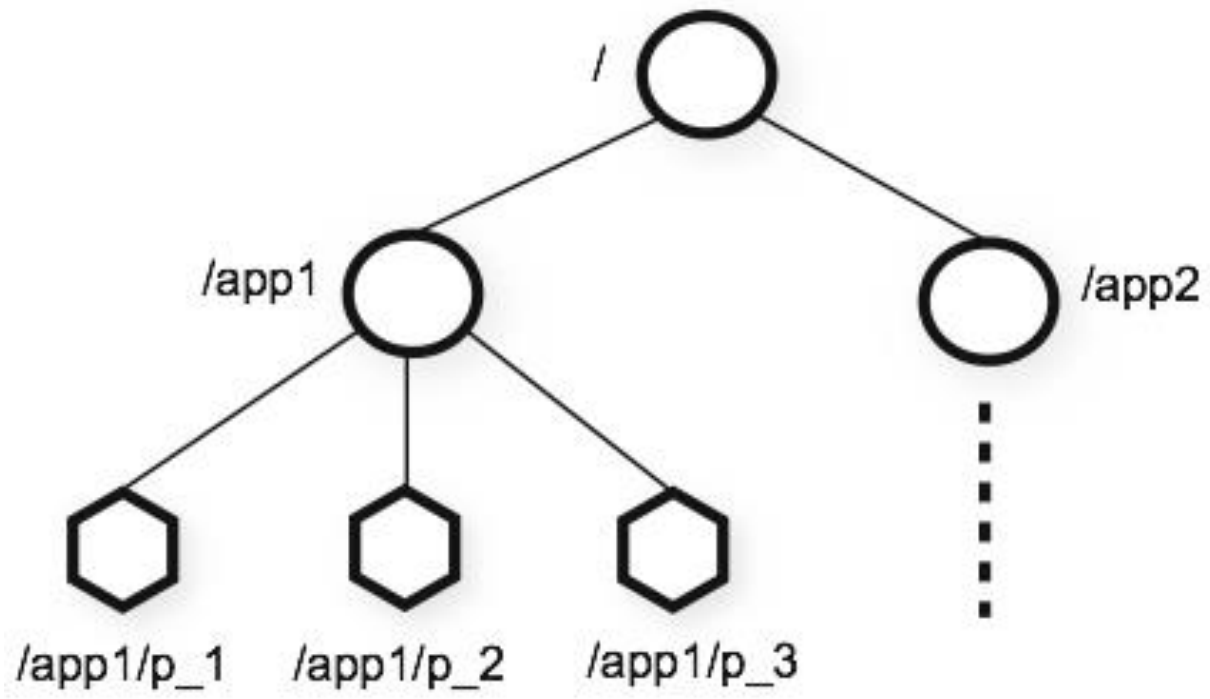
# zookeeper | 角色

| 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 |

# zookeeper | 数据模型

- Znode有两种类型，短暂的（ ephemeral）和持久的（ persistent）；
- Znode的类型在创建时确定并且之后不能再修改；
- 短暂znode的客户端会话结束时， zookeeper会将该短暂znode删除，短暂znode不可以有子节点；
- 持久znode不依赖于客户端会话，只有当客户端明确要删除该持久znode时才会被删除；
- Znode有四种形式的目录节点， PERSISTENT、PERSISTENT_SEQUENTIAL、EPHEMERAL、EPHEMERAL_SEQUENTIAL。

# zookeeper | 数据模型



# zookeeper | 集群安装

- 1、解压
- tar -zxvf zookeeper-3.4.6.tar.gz

- 2、复制示例文件
- cp /usr/local/zookeeper-3.4.6/conf/zoo_sample.cfg /usr/local/zookeeper-3.4.6/conf/zoo.cfg

- 3、修改zoo.cfg
- 修改数据存放目录：dataDir=/usr/local/zookeeper-3.4.6/data
- 配置三台zk服务器
- server.1=itr-mastertest01:2888:3888
- server.2=itr-mastertest02:2888:3888
- server.3=itr-nodetest01:2888:3888

- 4、创建data文件夹
- mkdir /usr/local/zookeeper-3.4.6/data

- 5、进入data目录
- vi myid -->内容为1
- 有表示符1,代表第server.1台server

- 6、scp zookeeper
- scp -rq zookeeper-3.4.6 itr-mastertest02:/usr/local/
- scp -rq zookeeper-3.4.6 itr-nodetest01:/usr/local/


- 7、修改myid文件
- [root@itr-mastertest02 data]# vi myid [值为2] 或者echo 2 > /usr/local/zookeeper-3.4.6/data/myid
- [root@itr-nodetest01 data]# vi myid [值为3]

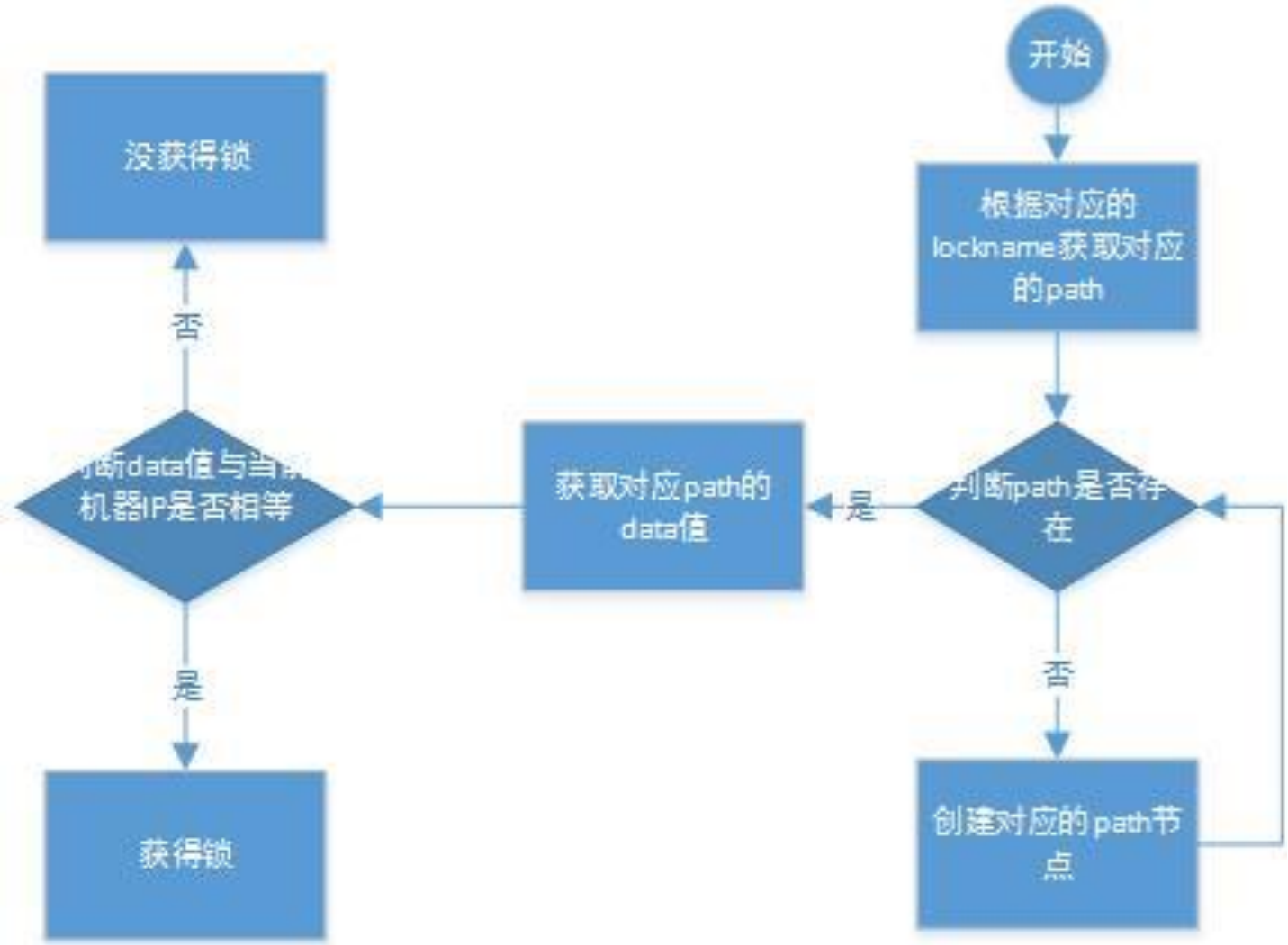
- 8、启动执行zk
- [root@itr-mastertest01 local]# zk zookeeper-3.4.6/bin/zkServer.sh start
- [root@itr-mastertest02 local]# zk zookeeper-3.4.6/bin/zkServer.sh start
- [root@itr-nodetest01 local]# zookeeper-3.4.6/bin/zkServer.sh start
- 9、查看zk状态
- [root@itr-mastertest01 local]# zookeeper-3.4.6/bin/zkServer.sh status
- 10、进入zk客户端
- [root@itr-mastertest01 local]# zookeeper-3.4.6/bin/zkCli.sh
- [zk: localhost:2181(CONNECTED) 3] get /zookeeper/quota [代表节点，每个节点存放数据信息]

# zookeeper | Observer配置

修改**zoo.cfg**中的两个配置：

- **peerType=observer**
- **server.1:localhost:2181:3181:observer**

# zookeeper | 分布式锁



# 大数据

# Thank you

提问时间？

**Blog： http://www.itweet.cn**

**PPT： https://github.com/itweet/course**

**Video： http://www.tudou.com/home/sparkjvm/**