大数据技术之Zookeeper

版本：V2.0

# 第1章 Zookeeper入门

## 1.1 概述

Zookeeper是一个开源的分布式的，为分布式应用提供协调服务的Apache项目。

1

## 1.2 特点



## 1.3 数据结构



## 1.4 应用场景

提供的服务包括：统一命名服务、统一配置管理、统一集群管理、服务器节点动态上下线、软负载均衡等。









## 1.5 下载地址

1．官网首页：

https://zookeeper.apache.org/

2．下载截图，如图5-5，5-6，5-7所示

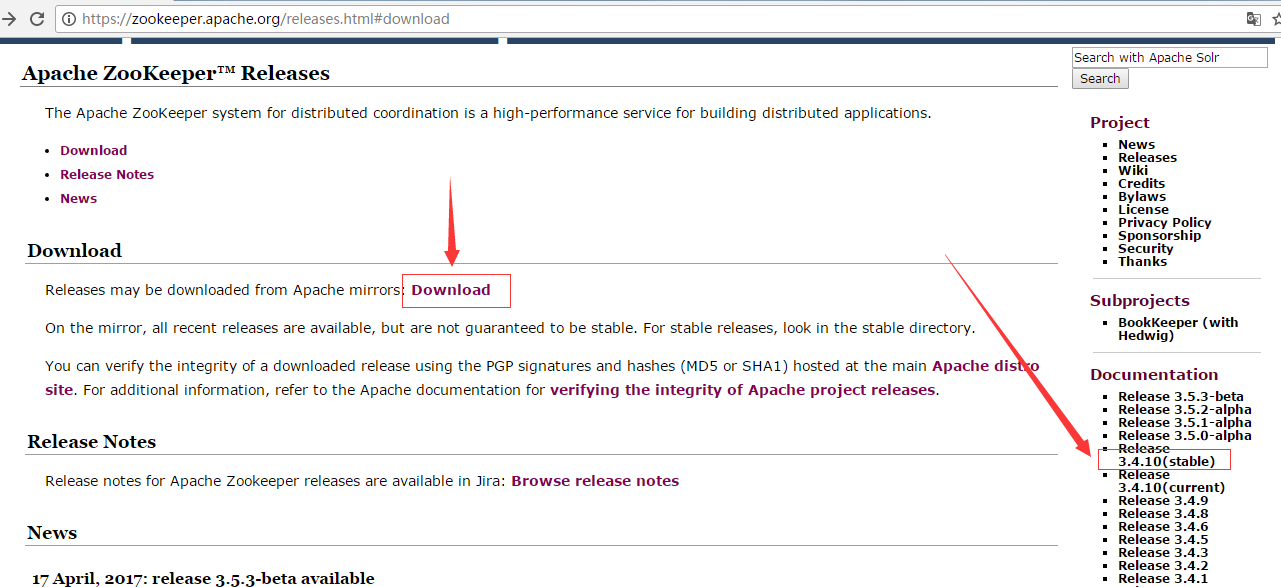


图5-5 Zookeeper下载（一）



图5-6 Zookeeper下载（二）

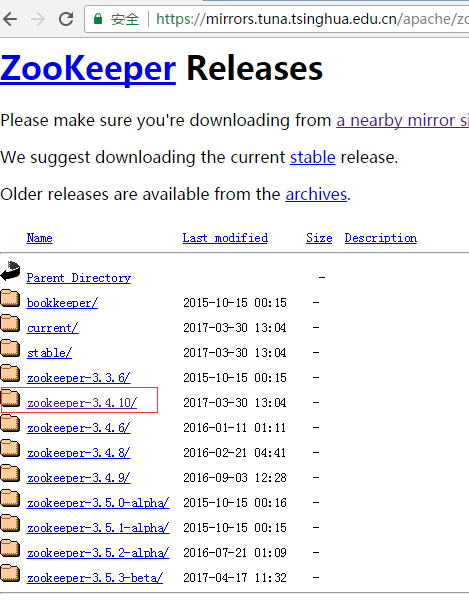


图5-7 Zookeeper下载（三）

# 第2章 Zookeeper安装

## 2.1 本地模式安装部署

1．安装前准备

（1）安装Jdk

（2）拷贝Zookeeper安装包到Linux系统下

（3）解压到指定目录

[atguigu@hadoop102 software]$ tar -zxvf zookeeper-3.5.7.tar.gz -C /opt/module/

2．配置修改

（1）将/opt/module/zookeeper-3.5.7/conf这个路径下的zoo\_sample.cfg修改为zoo.cfg；

[atguigu@hadoop102 conf]$ mv zoo\_sample.cfg zoo.cfg

（2）打开zoo.cfg文件，修改dataDir路径：

[atguigu@hadoop102 zookeeper-3.5.7]$ vim zoo.cfg

修改如下内容：

dataDir=/opt/module/zookeeper-3.5.7/zkData

（3）在/opt/module/zookeeper-3.5.7/这个目录上创建zkData文件夹

[atguigu@hadoop102 zookeeper-3.5.7]$ mkdir zkData

3．操作Zookeeper

（1）启动Zookeeper

[atguigu@hadoop102 zookeeper-3.5.7]$ bin/zkServer.sh start

（2）查看进程是否启动

[atguigu@hadoop102 zookeeper-3.5.7]$ jps

4020 Jps

4001 QuorumPeerMain

（3）查看状态：

[atguigu@hadoop102 zookeeper-3.5.7]$ bin/zkServer.sh status

ZooKeeper JMX enabled by default

Using config: /opt/module/zookeeper-3.5.7/bin/../conf/zoo.cfg

Mode: standalone

（4）启动客户端：

[atguigu@hadoop102 zookeeper-3.5.7]$ bin/zkCli.sh

（5）退出客户端：

[zk: localhost:2181(CONNECTED) 0] quit

（6）停止Zookeeper

[atguigu@hadoop102 zookeeper-3.5.7]$ bin/zkServer.sh stop

## 2.2 配置参数解读

Zookeeper中的配置文件zoo.cfg中参数含义解读如下：

1．tickTime =2000：通信心跳数，Zookeeper服务器与客户端心跳时间，单位毫秒

Zookeeper使用的基本时间，服务器之间或客户端与服务器之间维持心跳的时间间隔，也就是每个tickTime时间就会发送一个心跳，时间单位为毫秒。

它用于心跳机制，并且设置最小的session超时时间为两倍心跳时间。(session的最小超时时间是2\*tickTime)

2．initLimit =10：LF初始通信时限

集群中的Follower跟随者服务器与Leader领导者服务器之间初始连接时能容忍的最多心跳数（tickTime的数量），用它来限定集群中的Zookeeper服务器连接到Leader的时限。

3．syncLimit =5：LF同步通信时限

集群中Leader与Follower之间的最大响应时间单位，假如响应超过syncLimit \* tickTime，Leader认为Follwer死掉，从服务器列表中删除Follwer。

4．dataDir：数据文件目录+数据持久化路径

主要用于保存Zookeeper中的数据。

5．clientPort =2181：客户端连接端口

监听客户端连接的端口。

# 第3章 Zookeeper实战（开发重点）

## 3.1 分布式安装部署

1．集群规划

在hadoop102、hadoop103和hadoop104三个节点上部署Zookeeper。

2．解压安装

（1）解压Zookeeper安装包到/opt/module/目录下

[atguigu@hadoop102 software]$ tar -zxvf zookeeper-3.5.7.tar.gz -C /opt/module/

（2）同步/opt/module/zookeeper-3.5.7目录内容到hadoop103、hadoop104

[atguigu@hadoop102 module]$ xsync zookeeper-3.5.7/

3．配置服务器编号

（1）在/opt/module/zookeeper-3.5.7/这个目录下创建zkData

[atguigu@hadoop102 zookeeper-3.5.7]$ mkdir -p zkData

（2）在/opt/module/zookeeper-3.5.7/zkData目录下创建一个myid的文件

[atguigu@hadoop102 zkData]$ touch myid

添加myid文件，注意一定要在linux里面创建，在notepad++里面很可能乱码

（3）编辑myid文件

[atguigu@hadoop102 zkData]$ vi myid

在文件中添加与server对应的编号：

2

（4）拷贝配置好的zookeeper到其他机器上

[atguigu@hadoop102 zkData]$ xsync myid

并分别在hadoop103、hadoop104上修改myid文件中内容为3、4

4．配置zoo.cfg文件

（1）重命名/opt/module/zookeeper-3.5.7/conf这个目录下的zoo\_sample.cfg为zoo.cfg

[atguigu@hadoop102 conf]$ mv zoo\_sample.cfg zoo.cfg

（2）打开zoo.cfg文件

[atguigu@hadoop102 conf]$ vim zoo.cfg

修改数据存储路径配置

dataDir=/opt/module/zookeeper-3.5.7/zkData

增加如下配置

#######################cluster##########################

server.2=hadoop102:2888:3888

server.3=hadoop103:2888:3888

server.4=hadoop104:2888:3888

（3）同步zoo.cfg配置文件

[atguigu@hadoop102 conf]$ xsync zoo.cfg

（4）配置参数解读

server.A=B:C:D。

**A**是一个数字，表示这个是第几号服务器；

集群模式下配置一个文件myid，这个文件在dataDir目录下，这个文件里面有一个数据就是A的值，Zookeeper启动时读取此文件，拿到里面的数据与zoo.cfg里面的配置信息比较从而判断到底是哪个server。

**B**是这个服务器的地址；

**C**是这个服务器Follower与集群中的Leader服务器交换信息的端口；

**D**是万一集群中的Leader服务器挂了，需要一个端口来重新进行选举，选出一个新的Leader，而这个端口就是用来执行选举时服务器相互通信的端口。

4．集群操作

（1）分别启动Zookeeper

[atguigu@hadoop102 zookeeper-3.5.7]$ bin/zkServer.sh start

[atguigu@hadoop103 zookeeper-3.5.7]$ bin/zkServer.sh start

[atguigu@hadoop104 zookeeper-3.5.7]$ bin/zkServer.sh start

（2）查看状态

[atguigu@hadoop102 zookeeper-3.5.7]# bin/zkServer.sh status

JMX enabled by default

Using config: /opt/module/zookeeper-3.5.7/bin/../conf/zoo.cfg

Mode: follower

[atguigu@hadoop103 zookeeper-3.5.7]# bin/zkServer.sh status

JMX enabled by default

Using config: /opt/module/zookeeper-3.5.7/bin/../conf/zoo.cfg

Mode: leader

[atguigu@hadoop104 zookeeper-3.4.5]# bin/zkServer.sh status

JMX enabled by default

Using config: /opt/module/zookeeper-3.5.7/bin/../conf/zoo.cfg

Mode: follower

## 3.2 客户端命令行操作

表5-1

|  |  |
| --- | --- |
| 命令基本语法 | 功能描述 |
| help | 显示所有操作命令 |
| ls path [watch] | 使用 ls 命令来查看当前znode中所包含的内容 |
| ls2 path [watch] | 查看当前节点数据并能看到更新次数等数据 |
| create | 普通创建  -s 含有序列  -e 临时（重启或者超时消失） |
| get path [watch] | 获得节点的值 |
| set | 设置节点的具体值 |
| stat | 查看节点状态 |
| delete | 删除节点 |
| rmr | 递归删除节点 |

1．启动客户端

[atguigu@hadoop103 zookeeper-3.5.7]$ bin/zkCli.sh

2．显示所有操作命令

[zk: localhost:2181(CONNECTED) 1] help

3．查看当前znode中所包含的内容

[zk: localhost:2181(CONNECTED) 0] ls /

[zookeeper]

4．查看当前节点详细数据

[zk: localhost:2181(CONNECTED) 1] ls2 /

[zookeeper]

cZxid = 0x0

ctime = Thu Jan 01 08:00:00 CST 1970

mZxid = 0x0

mtime = Thu Jan 01 08:00:00 CST 1970

pZxid = 0x0

cversion = -1

dataVersion = 0

aclVersion = 0

ephemeralOwner = 0x0

dataLength = 0

numChildren = 1

5．分别创建2个普通节点

[zk: localhost:2181(CONNECTED) 3] create /sanguo "jinlian"

Created /sanguo

[zk: localhost:2181(CONNECTED) 4] create /sanguo/shuguo "liubei"

Created /sanguo/shuguo

6．获得节点的值

[zk: localhost:2181(CONNECTED) 5] get /sanguo

jinlian

cZxid = 0x100000003

ctime = Wed Aug 29 00:03:23 CST 2018

mZxid = 0x100000003

mtime = Wed Aug 29 00:03:23 CST 2018

pZxid = 0x100000004

cversion = 1

dataVersion = 0

aclVersion = 0

ephemeralOwner = 0x0

dataLength = 7

numChildren = 1

[zk: localhost:2181(CONNECTED) 6]

[zk: localhost:2181(CONNECTED) 6] get /sanguo/shuguo

liubei

cZxid = 0x100000004

ctime = Wed Aug 29 00:04:35 CST 2018

mZxid = 0x100000004

mtime = Wed Aug 29 00:04:35 CST 2018

pZxid = 0x100000004

cversion = 0

dataVersion = 0

aclVersion = 0

ephemeralOwner = 0x0

dataLength = 6

numChildren = 0

7．创建短暂节点

[zk: localhost:2181(CONNECTED) 7] create -e /sanguo/wuguo "zhouyu"

Created /sanguo/wuguo

（1）在当前客户端是能查看到的

[zk: localhost:2181(CONNECTED) 3] ls /sanguo

[wuguo, shuguo]

（2）退出当前客户端然后再重启客户端

[zk: localhost:2181(CONNECTED) 12] quit

[atguigu@hadoop104 zookeeper-3.5.7]$ bin/zkCli.sh

（3）再次查看根目录下短暂节点已经删除

[zk: localhost:2181(CONNECTED) 0] ls /sanguo

[shuguo]

8．创建带序号的节点

（1）先创建一个普通的根节点/sanguo/weiguo

[zk: localhost:2181(CONNECTED) 1] create /sanguo/weiguo "caocao"

Created /sanguo/weiguo

（2）创建带序号的节点

[zk: localhost:2181(CONNECTED) 2] create -s /sanguo/weiguo/xiaoqiao "jinlian"

Created /sanguo/weiguo/xiaoqiao0000000000

[zk: localhost:2181(CONNECTED) 3] create -s /sanguo/weiguo/daqiao "jinlian"

Created /sanguo/weiguo/daqiao0000000001

[zk: localhost:2181(CONNECTED) 4] create -s /sanguo/weiguo/diaocan "jinlian"

Created /sanguo/weiguo/diaocan0000000002

如果原来没有序号节点，序号从0开始依次递增。如果原节点下已有2个节点，则再排序时从2开始，以此类推。

9．修改节点数据值

[zk: localhost:2181(CONNECTED) 6] set /sanguo/weiguo "simayi"

10．节点的值变化监听

（1）在hadoop104主机上注册监听/sanguo节点数据变化

[zk: localhost:2181(CONNECTED) 26] [zk: localhost:2181(CONNECTED) 8] get /sanguo watch

（2）在hadoop103主机上修改/sanguo节点的数据

[zk: localhost:2181(CONNECTED) 1] set /sanguo "xisi"

（3）观察hadoop104主机收到数据变化的监听

WATCHER::

WatchedEvent state:SyncConnected type:NodeDataChanged path:/sanguo

11．节点的子节点变化监听（路径变化）

（1）在hadoop104主机上注册监听/sanguo节点的子节点变化

[zk: localhost:2181(CONNECTED) 1] ls /sanguo watch

[aa0000000001, server101]

（2）在hadoop103主机/sanguo节点上创建子节点

[zk: localhost:2181(CONNECTED) 2] create /sanguo/jin "simayi"

Created /sanguo/jin

（3）观察hadoop104主机收到子节点变化的监听

WATCHER::

WatchedEvent state:SyncConnected type:NodeChildrenChanged path:/sanguo

12．删除节点

[zk: localhost:2181(CONNECTED) 4] delete /sanguo/jin

13．递归删除节点

[zk: localhost:2181(CONNECTED) 15] rmr /sanguo/shuguo

14．查看节点状态

[zk: localhost:2181(CONNECTED) 17] stat /sanguo

cZxid = 0x100000003

ctime = Wed Aug 29 00:03:23 CST 2018

mZxid = 0x100000011

mtime = Wed Aug 29 00:21:23 CST 2018

pZxid = 0x100000014

cversion = 9

dataVersion = 1

aclVersion = 0

ephemeralOwner = 0x0

dataLength = 4

numChildren = 1

## 3.3 API应用

### 3.3.1 Eclipse环境搭建

1．创建一个Maven工程

2．添加pom文件

|  |
| --- |
| <dependencies>  <dependency>  <groupId>junit</groupId>  <artifactId>junit</artifactId>  <version>RELEASE</version>  </dependency>  <dependency>  <groupId>org.apache.logging.log4j</groupId>  <artifactId>log4j-core</artifactId>  <version>2.8.2</version>  </dependency>  <!-- https://mvnrepository.com/artifact/org.apache.zookeeper/zookeeper -->  <dependency>  <groupId>org.apache.zookeeper</groupId>  <artifactId>zookeeper</artifactId>  <version>3.5.7</version>  </dependency>  </dependencies> |

3．拷贝log4j.properties文件到项目根目录

需要在项目的src/main/resources目录下，新建一个文件，命名为“log4j.properties”，在文件中填入。

log4j.rootLogger=INFO, stdout

log4j.appender.stdout=org.apache.log4j.ConsoleAppender

log4j.appender.stdout.layout=org.apache.log4j.PatternLayout

log4j.appender.stdout.layout.ConversionPattern=%d %p [%c] - %m%n

log4j.appender.logfile=org.apache.log4j.FileAppender

log4j.appender.logfile.File=target/spring.log

log4j.appender.logfile.layout=org.apache.log4j.PatternLayout

log4j.appender.logfile.layout.ConversionPattern=%d %p [%c] - %m%n

### 3.3.2 创建ZooKeeper客户端

|  |
| --- |
| private static String connectString =  "hadoop102:2181,hadoop103:2181,hadoop104:2181";  private static int sessionTimeout = 2000;  private ZooKeeper zkClient = null;  @Before  public void init() throws Exception {  zkClient = new ZooKeeper(connectString, sessionTimeout, new Watcher() {  @Override  public void process(WatchedEvent event) {  // 收到事件通知后的回调函数（用户的业务逻辑）  System.out.println(event.getType() + "--" + event.getPath());  // 再次启动监听  try {  zkClient.getChildren("/", true);  } catch (Exception e) {  e.printStackTrace();  }  }  });  } |

### 3.3.3 创建子节点

|  |
| --- |
| // 创建子节点  @Test  public void create() throws Exception {  // 参数1：要创建的节点的路径； 参数2：节点数据 ； 参数3：节点权限 ；参数4：节点的类型  String nodeCreated = zkClient.create("/atguigu", "jinlian".getBytes(), Ids.OPEN\_ACL\_UNSAFE, CreateMode.PERSISTENT);  } |

### 3.3.4 获取子节点并监听节点变化

|  |
| --- |
| // 获取子节点  @Test  public void getChildren() throws Exception {  List<String> children = zkClient.getChildren("/", true);  for (String child : children) {  System.out.println(child);  }  // 延时阻塞  Thread.sleep(Long.MAX\_VALUE);  } |

### 3.3.5 判断Znode是否存在

|  |
| --- |
| // 判断znode是否存在  @Test  public void exist() throws Exception {  Stat stat = zkClient.exists("/eclipse", false);  System.out.println(stat == null ? "not exist" : "exist");  } |

## 3.4 监听服务器节点动态上下线案例（扩展）

1．需求

某分布式系统中，主节点可以有多台，可以动态上下线，任意一台客户端都能实时感知到主节点服务器的上下线。

2．需求分析，如图5-12所示



图5-12 服务器动态上下线

3．具体实现

（0）先在集群上创建/servers节点

[zk: localhost:2181(CONNECTED) 10] create /servers "servers"

Created /servers

（1）服务器端向Zookeeper注册代码

|  |
| --- |
| package com.atguigu.zkcase;  import java.io.IOException;  import org.apache.zookeeper.CreateMode;  import org.apache.zookeeper.WatchedEvent;  import org.apache.zookeeper.Watcher;  import org.apache.zookeeper.ZooKeeper;  import org.apache.zookeeper.ZooDefs.Ids;  public class DistributeServer {  private static String connectString = "hadoop102:2181,hadoop103:2181,hadoop104:2181";  private static int sessionTimeout = 2000;  private ZooKeeper zk = null;  private String parentNode = "/servers";    // 创建到zk的客户端连接  public void getConnect() throws IOException{    zk = new ZooKeeper(connectString, sessionTimeout, new Watcher() {  @Override  public void process(WatchedEvent event) {  }  });  }    // 注册服务器  public void registServer(String hostname) throws Exception{  String create = zk.create(parentNode + "/server", hostname.getBytes(), Ids.OPEN\_ACL\_UNSAFE, CreateMode.EPHEMERAL\_SEQUENTIAL);    System.out.println(hostname +" is online "+ create);  }    // 业务功能  public void business(String hostname) throws Exception{  System.out.println(hostname+" is working ...");    Thread.sleep(Long.MAX\_VALUE);  }    public static void main(String[] args) throws Exception {    // 1获取zk连接  DistributeServer server = new DistributeServer();  server.getConnect();    // 2 利用zk连接注册服务器信息  server.registServer(args[0]);    // 3 启动业务功能  server.business(args[0]);  }  } |

（2）客户端代码

|  |
| --- |
| package com.atguigu.zkcase;  import java.io.IOException;  import java.util.ArrayList;  import java.util.List;  import org.apache.zookeeper.WatchedEvent;  import org.apache.zookeeper.Watcher;  import org.apache.zookeeper.ZooKeeper;  public class DistributeClient {  private static String connectString = "hadoop102:2181,hadoop103:2181,hadoop104:2181";  private static int sessionTimeout = 2000;  private ZooKeeper zk = null;  private String parentNode = "/servers";  // 创建到zk的客户端连接  public void getConnect() throws IOException {  zk = new ZooKeeper(connectString, sessionTimeout, new Watcher() {  @Override  public void process(WatchedEvent event) {  // 再次启动监听  try {  getServerList();  } catch (Exception e) {  e.printStackTrace();  }  }  });  }  // 获取服务器列表信息  public void getServerList() throws Exception {    // 1获取服务器子节点信息，并且对父节点进行监听  List<String> children = zk.getChildren(parentNode, true);  // 2存储服务器信息列表  ArrayList<String> servers = new ArrayList<>();    // 3遍历所有节点，获取节点中的主机名称信息  for (String child : children) {  byte[] data = zk.getData(parentNode + "/" + child, false, null);  servers.add(new String(data));  }  // 4打印服务器列表信息  System.out.println(servers);  }  // 业务功能  public void business() throws Exception{  System.out.println("client is working ...");  Thread.sleep(Long.MAX\_VALUE);  }  public static void main(String[] args) throws Exception {  // 1获取zk连接  DistributeClient client = new DistributeClient();  client.getConnect();  // 2获取servers的子节点信息，从中获取服务器信息列表  client.getServerList();  // 3业务进程启动  client.business();  }  } |

# 第4章 Zookeeper内部原理

## 4.1 节点类型



## 4.2 Stat结构体

1）czxid-创建节点的事务zxid

每次修改ZooKeeper状态都会收到一个zxid形式的时间戳，也就是ZooKeeper事务ID。

事务ID是ZooKeeper中所有修改总的次序。每个修改都有唯一的zxid，如果zxid1小于zxid2，那么zxid1在zxid2之前发生。

2）ctime - znode被创建的毫秒数(从1970年开始)

3）mzxid - znode最后更新的事务zxid

4）mtime - znode最后修改的毫秒数(从1970年开始)

5）pZxid-znode最后更新的子节点zxid

6）cversion - znode子节点变化号，znode子节点修改次数

7）dataversion - znode数据变化号

8）aclVersion - znode访问控制列表的变化号

9）ephemeralOwner- 如果是临时节点，这个是znode拥有者的session id。如果不是临时节点则是0。

10）dataLength- znode的数据长度

11）numChildren - znode子节点数量

## 4.3 监听器原理（面试重点）



图5-10 监听器原理

## 4.4 Paxos算法（扩展）

Paxos算法一种基于消息传递且具有高度容错特性的一致性算法。

分布式系统中的节点通信存在两种模型：共享内存（Shared memory）和消息传递（Messages passing）。基于消息传递通信模型的分布式系统，不可避免的会发生以下错误：进程可能会慢、被杀死或者重启，消息可能会延迟、丢失、重复，在基础 Paxos 场景中，先不考虑可能出现消息篡改即拜占庭错误的情况。Paxos 算法解决的问题是在一个可能发生上述异常的分布式系统中如何就某个值达成一致，保证不论发生以上任何异常，都不会破坏决议的一致性。



Paxos算法流程中的每条消息描述如下：

1. Prepare: Proposer生成全局唯一且递增的Proposal ID (可使用时间戳加Server ID)，向所有Acceptors发送Prepare请求，这里无需携带提案内容，只携带Proposal ID即可。
2. Promise: Acceptors收到Prepare请求后，做出“两个承诺，一个应答”。

两个承诺：

1. 不再接受Proposal ID小于等于（注意：这里是<= ）当前请求的Prepare请求。
2. 不再接受Proposal ID小于（注意：这里是< ）当前请求的Propose请求。

一个应答：

1. 不违背以前做出的承诺下，回复已经Accept过的提案中Proposal ID最大的那个提案的Value和Proposal ID，没有则返回空值。
2. Propose: Proposer 收到多数Acceptors的Promise应答后，从应答中选择Proposal ID最大的提案的Value，作为本次要发起的提案。如果所有应答的提案Value均为空值，则可以自己随意决定提案Value。然后携带当前Proposal ID，向所有Acceptors发送Propose请求。
3. Accept: Acceptor收到Propose请求后，在不违背自己之前做出的承诺下，接受并持久化当前Proposal ID和提案Value。
4. Learn: Proposer收到多数Acceptors的Accept后，决议形成，将形成的决议发送给所有Learners。

下面我们针对上述描述做三种情况的推演举例：为了简化流程，我们这里不设置Learner。





Paxos算法缺陷：在网络复杂的情况下，一个应用Paxos算法的分布式系统，可能很久无法收敛，甚至陷入活锁的情况。



造成这种情况的原因是系统中有一个以上的Proposer，多个Proposers相互争夺Acceptors，造成迟迟无法达成一致的情况。针对这种情况，一种改进的Paxos算法被提出：从系统中选出一个节点作为Leader，只有Leader能够发起提案。这样，一次Paxos流程中只有一个Proposer，不会出现活锁的情况，此时只会出现例子中第一种情况。

## 4.5 选举机制（面试重点）

1）半数机制：集群中半数以上机器存活，集群可用。所以Zookeeper适合安装奇数台服务器。

2）Zookeeper虽然在配置文件中并没有指定Master和Slave。但是，Zookeeper工作时，是有一个节点为Leader，其他则为Follower，Leader是通过内部的选举机制临时产生的。

3）以一个简单的例子来说明整个选举的过程。

假设有五台服务器组成的Zookeeper集群，它们的id从1-5，同时它们都是最新启动的，也就是没有历史数据，在存放数据量这一点上，都是一样的。假设这些服务器依序启动，来看看会发生什么，如图5-8所示。



图5-8 Zookeeper的选举机制

（1）服务器1启动，发起一次选举。服务器1投自己一票。此时服务器1票数一票，不够半数以上（3票），选举无法完成，服务器1状态保持为LOOKING；

（2）服务器2启动，再发起一次选举。服务器1和2分别投自己一票并交换选票信息：此时服务器1发现服务器2的ID比自己目前投票推举的（服务器1）大，更改选票为推举服务器2。此时服务器1票数0票，服务器2票数2票，没有半数以上结果，选举无法完成，服务器1，2状态保持LOOKING

（3）服务器3启动，发起一次选举。此时服务器1和2都会更改选票为服务器3。此次投票结果：服务器1为0票，服务器2为0票，服务器3为3票。此时服务器3的票数已经超过半数，服务器3当选Leader。服务器1，2更改状态为FOLLOWING，服务器3更改状态为LEADING；

（4）服务器4启动，发起一次选举。此时服务器1，2，3已经不是LOOKING状态，不会更改选票信息。交换选票信息结果：服务器3为3票，服务器4为1票。此时服务器4服从多数，更改选票信息为服务器3，并更改状态为FOLLOWING；

（5）服务器5启动，同4一样当小弟。

## 4.6 写数据流程



# 第5章 企业面试真题

## 5.1 请简述ZooKeeper的选举机制

详见4.4。

## 5.2 ZooKeeper的监听原理是什么？

详见4.3。

## 5.3 ZooKeeper的部署方式有哪几种？集群中的角色有哪些？集群最少需要几台机器？

（1）部署方式单机模式、集群模式

（2）角色：Leader和Follower

（3）集群最少需要机器数：3

## 5.4 ZooKeeper的常用命令

ls create get delete set…