3，启动

zkServer.sh start

QuorumPeerMain:

SnapShot.Class：Interface, {

long deserialize() //从最新的快照中反序列化出一个数据树，并且返回最后的 zxid

void serialize() //将数据树和session持久化存储

File findMostRecentSnapshot() //寻找最近的快照

void close() //关闭快照释放资源

}

TxnHeader: Class{

private long clientId; //用户ID

private int cxid;

private long zxid;

private long time;

private int type;

getXXX()

setXXX()

}

TxnLog:Class{

void rollLog()

boolean append(TxnHeader hdr, Record r) // Append a request to the transaction log

TxnIterator read(long zxid) throws IOException; //start reading the transaction logs（zxid）

long getLastLoggedZxid() // the last zxid of the logged transactions

boolean truncate(long zxid)

long getDbId() // the DbId for this transaction log

void commit() // commit the transaction and make sure they are persisted

void close() // close the transaction log

//

public interface TxnIterator {

// return the transaction header

TxnHeader getHeader();

// return the transaction record

Record getTxn();

// go to the next transaction record

boolean next()

//close files and release the resources

Void close()

long getStorageSize() throws IOException;

}

}

Vote:Class{

Vote(); // 更新这台服务器的投票信息

}

public enum ServerState {

LOOKING, FOLLOWING, LEADING, OBSERVING;

}

史诗级详解：

**Code1： [QuorumPeerMain.Class]**

public static void main(String[] args) {

//初始化

QuorumPeerMain main = new QuorumPeerMain();

try {

//初始化并且运行

main.initializeAndRun(args);

}

}

**Code2: [QuorumPeerMain.Class]**

protected void initializeAndRun(String[] args)

throws ConfigException, IOException {

//集群初始化配置

QuorumPeerConfig config = new QuorumPeerConfig();

if (args.length == 1) {

//从配置文件中读取信息

config.parse(args[0]);

}

if (args.length == 1 && config.servers.size() > 0) {

//运行集群模式

runFromConfig(config);

} else {

// there is only server in the quorum -- run as standalone

//运行单机模式

ZooKeeperServerMain.main(args);

}

}

**Code3: [QuorumPeerConfig.Class]**

public void parse(String path) throws ConfigException {

try {

//读入配置文件

File configFile = (new VerifyingFileFactory.Builder(LOG)

.warnForRelativePath()

.failForNonExistingPath()

.build()).create(path);

Properties cfg = new Properties();

FileInputStream in = new FileInputStream(configFile); // 读入文件流

try {

cfg.load(in); //将数据加载入cfg中

} finally {

in.close();

}

parseProperties(cfg); // 解析文件

configFileStr = path; //记录配置文件地址

}

}

**Code4: [QuorumPeerConfig.Class]**

public void parseProperties(Properties zkProp) {

int clientPort = 0;

String clientPortAddress = null;

VerifyingFileFactory vff = new VerifyingFileFactory.Builder(LOG).warnForRelativePath().build();

//读取配置并且设置配置，文件

for (Entry<Object, Object> entry : zkProp.entrySet()) {

String key = entry.getKey().toString().trim();

String value = entry.getValue().toString().trim();

LOG.info("properties:key=" +key+ "\* value="+value+"\*");

if (key.equals("dataDir")) {

dataDir = vff.create(value);

} else if (key.equals("dataLogDir")) {

dataLogDir = vff.create(value);

} else if (key.equals("clientPort")) {

clientPort = Integer.parseInt(value);

} else if(key.equals("startType")){

startType=Integer.parseInt(value);

} else if (key.equals("localSessionsEnabled")) {

localSessionsEnabled = Boolean.parseBoolean(value);

} else if (key.equals("localSessionsUpgradingEnabled")) {

localSessionsUpgradingEnabled = Boolean.parseBoolean(value);

} else if (key.equals("clientPortAddress")) {

clientPortAddress = value.trim();

}

……………

else {

System.setProperty("zookeeper." + key, value);

}

}

// 如果dataLogDir为空，则将dataLogDir=dataDir

if (dataLogDir == null) {

LOG.info("parseproperties datalogdir null");

dataLogDir = dataDir;

}

}

}

**Code5： runFromCongif() [QuorumPeerMain.Class]**

public void runFromConfig(QuorumPeerConfig config) throws IOException {

try {

ManagedUtil.registerLog4jMBeans();

} catch (JMException e) {

LOG.warn("Unable to register log4j JMX control", e);

}

LOG.info("Starting quorum peer");

try {

//创建NIOServerCnxnFactory实例

ServerCnxnFactory cnxnFactory = ServerCnxnFactory.createFactory();

// 初始化线程，配置监听，注册监听

cnxnFactory.configure(config.getClientPortAddress(),

config.getMaxClientCnxns());

quorumPeer = new QuorumPeer();

//设置集群鉴定端口

quorumPeer.setClientPortAddress(config.getClientPortAddress());

// 设置快照以及事务地址

quorumPeer.setTxnFactory(new FileTxnSnapLog(

new File(config.getDataLogDir()),

new File(config.getDataDir())));

//获得服务器集群

quorumPeer.setQuorumPeers(config.getServers());

// 选举算法

quorumPeer.setElectionType(config.getElectionAlg());

// 服务器ID

quorumPeer.setMyid(config.getServerId());

quorumPeer.setTickTime(config.getTickTime());

quorumPeer.setMinSessionTimeout(config.getMinSessionTimeout());

quorumPeer.setMaxSessionTimeout(config.getMaxSessionTimeout());

quorumPeer.setInitLimit(config.getInitLimit());

quorumPeer.setSyncLimit(config.getSyncLimit());

//投票决定方式，默认超过半数就通过

quorumPeer.setQuorumVerifier(config.getQuorumVerifier());

quorumPeer.setCnxnFactory(cnxnFactory);

// 建立ZK数据 ，包含有数据树等

quorumPeer.setZKDatabase(new ZKDatabase(quorumPeer.getTxnFactory()));

quorumPeer.setLearnerType(config.getPeerType());

quorumPeer.setSyncEnabled(config.getSyncEnabled());

quorumPeer.setQuorumListenOnAllIPs(config.getQuorumListenOnAllIPs());

//启动

quorumPeer.start();

quorumPeer.join();

} catch (InterruptedException e) {

// warn, but generally this is ok

LOG.warn("Quorum Peer interrupted", e);

}

}

当Leader选举完成后，先看Leader端操作

Run()[QuorumPeer.java]

case LEADING:

LOG.info("LEADING");

try {

setLeader(makeLeader(logFactory));

leader.lead();

setLeader(null);

} catch (Exception e) {

LOG.warn("Unexpected exception",e);

} finally {

if (leader != null) {

leader.shutdown("Forcing shutdown");

setLeader(null);

}

setPeerState(ServerState.LOOKING);

}

break;

}

Leader:[Leader.java]

Leader(QuorumPeer self,LeaderZooKeeperServer zk) throws IOException {

this.self = self;

try {

//集群模式监听接受服务信息端口

if (self.getQuorumListenOnAllIPs()) {

ss = new ServerSocket(self.getQuorumAddress().getPort());

} else {

ss = new ServerSocket();

}

ss.setReuseAddress(true);

if (!self.getQuorumListenOnAllIPs()) {

ss.bind(self.getQuorumAddress());

}

} catch (BindException e) {

if (self.getQuorumListenOnAllIPs()) {

LOG.error("Couldn't bind to port " + self.getQuorumAddress().getPort(), e);

} else {

LOG.error("Couldn't bind to " + self.getQuorumAddress(), e);

}

throw e;

}

this.zk=zk;

}