Zookeeper分布式系统开发实战 第9课

DATAGURU专业数据分析社区

法律声明

【声明】本视频和幻灯片为炼数成金网络课程的教学资料，所有资料只能在课程内使用，不得在课程以外范围散播，违者将可能被追究法律和经济责任。

课程详情访问炼数成金培训网站

http://edu.dataguru.cn

炼数成金逆向收费式网络课程

- Dataguru（炼数成金）是专业数据分析网站，提供教育，媒体，内容，社区，出版，数据分析业务等服务。我们的课程采用新兴的互联网教育形式，独创地发展了逆向收费式网络培训课程模式。既继承传统教育重学习氛围，重竞争压力的特点，同时又发挥互联网的威力打破时空限制，把天南地北志同道合的朋友组织在一起交流学习，使到原先孤立的学习个体组合成有组织的探索力量。并且把原先动辄成千上万的学习成本，直线下降至百元范围，造福大众。我们的目标是：低成本传播高价值知识，构架中国第一的网上知识流转阵地。
- 关于逆向收费式网络的详情，请看我们的培训网站 http://edu.dataguru.cn

第九讲 zk详细配置

- 日志配置
  - 基于log4j的日志输出
  - Windows默认的日志输出目录为安装目录，比如安装目录为G:\开发软件\zookeeper-3.4.6\，则日志输出目录就为G:\开发软件\zookeeper-3.4.6\zookeeper-3.4.6，默认日志文件名叫zookeeper.log
  - Linux默认的控制台输出为bin/zookeeper.out
  - 默认没有开启日志文件输出
  - 修改日志相关属性
    - conf/log4j.properties
    - bin/zkEnv.cmd或者linux下的zkEnv.sh
      - set ZOO_LOG_DIR=G:\\datadir //log4j里面配置的不生效
      - 增加配置属性ZOO_LOG_DIR

第九讲 zk详细配置

- 其它配置
  - 两种配置形式
    - 基于java的系统属性配置，比如：-Djava.library.path
    - Zk自身的zoo.conf文件

属性名称	作用	说明
dataLogDir	配置事务日志文件存储目录	1.不支持系统属性配置 2.默认为属性dataDir的值 3.在高并发下，有大量的事务日志和快照，会导致磁盘IO瓶颈，因此在高并发下，不建议使用默认配置，最好把dataDir和此属性配置的目录分在不同的磁盘下，从而提高IO
snapCount	两次快照间隔的事务日志条数	1.事务日志条数达到这个数目，就要触发数据快照 2.默认值为100000 3.仅支持系统属性配置方式

第九讲 zk详细配置

属性名称	作用	说明
preAllocSize	事务日志文件预分配的磁盘空间大小	1.仅支持系统属性配置，zookeeper.preAllocSize 2.默认值为65535，即64M 3.此参数与snapCount有关，snapCount大，就需要多分配
minSessionTimeout maxSessionTimeout	会话失效的时间的边界控制（服务器端）	1.不支持系统属性 2.默认为ticktime的2倍和20倍 3.当客户端传递过来的超时时间不在这两个参数之间，则最小取minSessionTimeout，最大取maxSessionTimeout
maxClientCnxns	从socket层限制客户端与单台服务器的并发连接数	1.不支持系统属性，默认值为60，0表示不限制 2.以IP地址为粒度进行控制 3.只能控制单台机器，不能控制总连接
Jute.maxbuffer	配置单个节点最大的数据大小	1.默认是10M，单位是字节，仅支持系统属性方式配置 2.Zk上存储的数据不宜过多，主要是考虑多节点写入的性能 3.需要在服务器端和客户端都配置才能生效

第九讲 zk详细配置

属性名称	作用	说明
Autopurge.snapRetainCount	自动清理快照和事务日志时需要保留的文件数	1.不支持系统属性配置，系统默认为3，可以不用配置 2.最小值为3，避免磁盘损坏后不能恢复数据
Autopurge.purgeInterval	自动清理快照和事务的周期	1.不支持系统属性，默认为0，表示不开启自动清理 2.与Autopurge.snapRetainCount属性一起配合使用 3.配置为负数也表示不清理
fsync.warningthresholdms	事务日志刷新到磁盘的报警阀值	1.支持系统属性，默认值为1000ms 2.如果fsync的操作超过此时间就会在日志中打印报警日志
forceSync	日志提交时是否强制刷磁盘	1.默认为true 2.仅支持系统属性配置:zookeeper.forceSync 3.如果设置为no，可以提升写入性能，但是会有数据丢失风险
cnxTimeout	选举过程中，服务器之间创建tcp连接的超时时间	1.仅支持系统属性配置：zokeeper.cnxTimeout 2.默认为5000ms

第九讲 四字命令

- 定义
  - 长度为4个英文字母的管理命令，比如stat就是其中一个
- 使用方式
  - telnet
    - telnet ip port
    - 命令执行
  - nc
    - echo 命令 |nc ip port

```
[root@localhost bin]# echo stat |nc localhost 2181
Zookeeper version: 3.4.6-1569965, built on 02/20/2014 09:09 GMT
Clients:
 /127.0.0.1:34066[0](queued=0,recved=1,sent=0)

Latency min/avg/max: 0/0/0
Received: 1
Sent: 0
Connections: 1
Outstanding: 0
Zxid: 0x0
Mode: standalone
Node count: 4
```

第九讲 四字命令

- nc
  - 介绍：一个简单、可靠的网络工具，可通过TCP或UDP协议传输读写数据
  - 安装：
    - 下载rpm：http://sourceforge.net/projects/netcat/files/netcat/0.7.1/netcat-0.7.1-1.i386.rpm/download
    - 安装依赖包：
      - 配置yum源，配置163的centos的yum源即可
      - 执行命令：yum install glibc.i686
    - 安装netcat： rpm –ivh netcat-0.7.1-1.i386.rpm

第九讲 四字命令

- conf
  - 用于输出基本配置信息，也可以查看某些运行时参数
  - telnet localhost 2181,然后执行conf
  - Echo conf |nc localhost 2181

集群环境的配置信息

```
[root@localhost bin]# echo conf |nc localhost 2181
clientPort=2181
dataDir=/tmp/zookeeper/version-2
dataLogDir=/tmp/zookeeper/version-2
tickTime=2000
maxClientCnxns=60
minSessionTimeout=4000
maxSessionTimeout=40000
serverId=0
[root@localhost bin]#
```

单机环境的配置信息

第九讲 四字命令

- cons
  - 用于输出当前客户端所有连接的详细信息，包括客户端ip 会话id等
  - telnet localhost 2181,然后执行cons
  - Echo cons |nc localhost 2181

```
/0:0:0:0:0:0:0:1:36602[0](queued=0,recved=1,sent=0)
/0:0:0:0:0:0:0:1:26094[1](queued=0,recved=796,sent=796,sid=0x152267f5e810000,lop=PING,
est=1452344842822,to=30000,lcxid=0x1,lzxid=0xffffffffffffffff,lresp=1452352788941,llat=1,
minlat=0,avglat=1,maxlat=54)
```

```
[root@localhost bin]# echo cons |nc localhost 2181
/127.0.0.1:34071[1](queued=0,recved=1,sent=1,sid=0x1522687d7bd0000,lop=SESS,est=1452348261921,to=30000,
lcxid=0x0,lzxid=0x1,lresp=1452348262110,llat=106,minlat=0,avglat=106,maxlat=106)
/127.0.0.1:34072[0](queued=0,recved=1,sent=0)
```

单机环境的配置信息

第九讲 四字命令

```
pwriter.print("[");
int interestOps = getInterestOps();
pwriter.print(interestOps == 0 ? "0" : Integer.toHexString(interestOps));
pwriter.print("](queued=");
pwriter.print(getOutstandingRequests());
pwriter.print(",recved=");
pwriter.print(getPacketsReceived());
pwriter.print(",sent=");
pwriter.print(getPacketsSent());

if (!brief) {
    long sessionId = getSessionId();
    if (sessionId != 0) {
        pwriter.print(",sid=0x");
        pwriter.print(Long.toHexString(sessionId));
        pwriter.print(",lop=");
        pwriter.print(getLastOperation());
        pwriter.print(",est=");
        pwriter.print(getEstablished().getTime());
        pwriter.print(",to=");
        pwriter.print(getSessionTimeout());
        long lastCxid = getLastCxid();
        if (lastCxid >= 0) {
            pwriter.print(",lcxid=0x");
            pwriter.print(Long.toHexString(lastCxid));
        }
        pwriter.print(",lzxid=0x");
        pwriter.print(Long.toHexString(getLastZxid()));
        pwriter.print(",lresp=");
        pwriter.print(getLastResponseTime());
        pwriter.print(",llat=");
        pwriter.print(getLastLatency());
        pwriter.print(",minlat=");
        pwriter.print(getMinLatency());
        pwriter.print(",avglat=");
        pwriter.print(getAvgLatency());
        pwriter.print(",maxlat=");
```

输出的信息都来自这段代码

第九讲 四字命令

- crst
  - 用于重置客户端连接统计信息
  - telnet localhost 2181,然后执行crst
  - echo crst |nc localhost 2181

```
[root@localhost bin]# echo cons |nc localhost 2181
 /127.0.0.1:34071[1](queued=0,recved=4191,sent=4191,sid=0x1522687d7bd0000,lop=PING,est=1452348261921,to=30000,lcxid=0x6,lzxid=0x4,lresp=1452390130101,llat=0,minlat=0,avglat=0,maxlat=106)
 /127.0.0.1:34129[0](queued=0,recved=1,sent=0)
```

Crst执行之前

```
[root@localhost bin]# echo crst |nc localhost 2181
Connection stats reset.
```

```
[root@localhost bin]# echo cons |nc localhost 2181
 /127.0.0.1:34131[0](queued=0,recved=1,sent=0)
 /127.0.0.1:34071[1](queued=0,recved=0,sent=0,sid=0x1522687d7bd0000,lop=NA,est=1452348261921,to=30000,lzxid=0xffffffffffffffff,lresp=0,llat=0,minlat=0,avglat=0,maxlat=0)
```

Crst执行之后

第九讲 四字命令

- dump
  - 用于输出当前集群的所有会话信息，包括会话id以及临时节点等信息
  - 如果dump的是leader节点，则还会有会话的超时时间
  - telnet localhost 2181,然后执行dump
  - echo dump |nc localhost 2181

在leader节点执行

```
    SessionTracker dump:
Session Sets (3):
0 expire at Sun Jan 10 10:09:54 CST 2016:
0 expire at Sun Jan 10 10:10:04 CST 2016:
1 expire at Sun Jan 10 10:10:14 CST 2016:
        0x152267f5e810000
ephemeral nodes dump:
Sessions with Ephemerals (1):
0x152267f5e810000:
        /enode
```

```
    SessionTracker dump:
org.apache.zookeeper.server.quorum.LearnerSessionTracker@7a7ff2aa
ephemeral nodes dump:
Sessions with Ephemerals (1):
0x152267f5e810000:
        /enode
```

在follower节点执行

第九讲 四字命令

- envi
  - 用于输出运行时的环境信息
  - telnet localhost 2181,然后执行envi
  - echo envi |nc localhost 2181

```
Environment:
zookeeper.version=3.4.6-1569965, built on 02/20/2014 09:09 GMT
host.name=2012-20140406IS
java.version=1.7.0_67
java.vendor=Oracle Corporation
java.home=D:\Program Files\Java\jdk1.7.0_67\jre
java.class.path=G:\开发软件\zookeeper-3.4.6\zookeeper-3.4.6 - 2\bin\..\build\classes;G:\开发软件\zookeeper-3.4.6\zookeeper-3
ookeeper-3.4.6\zookeeper-3.4.6 - 2\bin\..\lib\jline-0.9.94.jar;G:\开发软件\zookeeper-3.4.6\zookeeper-3.4.6 - 2\bin\..\lib\lo
r-3.4.6\zookeeper-3.4.6 - 2\bin\..\lib\slf4j-api-1.6.1.jar;G:\开发软件\zookeeper-3.4.6\zookeeper-3.4.6 - 2\bin\..\lib\slf4j-
java.library.path=D:\Program Files\Java\jdk1.7.0_67\bin;C:\Windows\Sun\Java\bin;C:\Windows\system32;C:\Windows;E:\apache-mav
s\System32\WindowsPowerShell\v1.0\;C:\Program Files (x86)\Windows7Master;D:\mysql-5.6.16-winx64\bin;C:\strawberry\c\bin;C:\s
ne extensions for PHP\;C:\Program Files\TortoiseSUN\bin;.;;.
java.io.tmpdir=C:\Users\ADMINI~1\AppData\Local\Temp\
java.compiler=<NA>
os.name=Windows 7
os.arch=amd64
os.version=6.1
user.name=Administrator
user.home=C:\Users\Administrator
user.dir=G:\开发软件\zookeeper-3.4.6\zookeeper-3.4.6 - 2\bin
```

第九讲 四字命令

- ruok
  - 用于输出当前zk服务器运行是否正常，仅仅代表2181端口和四字命令流程执行正常，不能完全代表zk运行正常，最有效的命令是stat
  - telnet localhost 2181,然后执行ruok
  - echo ruok |nc localhost 2181

```
[root@localhost bin]# echo ruok |nc localhost 2181
imok[root@localhost bin]#
```

第九讲 四字命令

- stat
  - 用于获取服务器端的运行状态：zk版本 打包信息 运行时角色 集群数据节点等
  - telnet localhost 2181,然后执行stat
  - echo stat |nc localhost 2181

```
Zookeeper version: 3.4.6-1569965, built on 02/20/2014 09:09 GMT
Clients:
 /0:0:0:0:0:0:0:1:37366[0](queued=0,recved=1,sent=0)
 /0:0:0:0:0:0:0:1:26094[1](queued=0,recved=4923,sent=4923)

Latency min/avg/max: 0/1/1527
Received: 4940
Sent: 4939
Connections: 2
Outstanding: 0
Zxid: 0x1600000006
Mode: follower
Node count: 43
```

```
[root@localhost bin]# echo stat |nc localhost 2181
Zookeeper version: 3.4.6-1569965, built on 02/20/2014 09:09 GMT
Clients:
 /127.0.0.1:34148[0](queued=0,recved=1,sent=0)
 /0:0:0:0:0:0:0:1:43654[1](queued=0,recved=1,sent=1)
 /0:0:0:0:0:0:0:1:43653[1](queued=0,recved=1,sent=1)

Latency min/avg/max: 0/0/106
Received: 4932
Sent: 4931
Connections: 3
Outstanding: 0
Zxid: 0x9
Mode: standalone
Node count: 6
```

第九讲 四字命令

输出的信息都来自这段代码

```
@Override
public String toString(){
    StringBuilder sb = new StringBuilder();
    sb.append("Latency min/avg/max: " + getMinLatency() + "/"
            + getAvgLatency() + "/" + getMaxLatency() + "\n");
    sb.append("Received: " + getPacketsReceived() + "\n");
    sb.append("Sent: " + getPacketsSent() + "\n");
    sb.append("Connections: " + getNumAliveClientConnections() + "\n");

    if (provider != null) {
        sb.append("Outstanding: " + getOutstandingRequests() + "\n");
        sb.append("Zxid: 0x"+ Long.toHexString(getLastProcessedZxid())+ "\n");
    }
    sb.append("Mode: " + getServerState() + "\n");
    return sb.toString();
}
```

```
// mutators
synchronized void updateLatency(long requestCreateTime) {
    long latency = System.currentTimeMillis() - requestCreateTime;
    totalLatency += latency;
    count++;
    if (latency < minLatency) {
        minLatency = latency;
    }
    if (latency > maxLatency) {
        maxLatency = latency;
    }
}
```

第九讲 四字命令

- srvr
  - 与stat功能类似，但是不输出连接信息
  - telnet localhost 2181,然后执行srvr
  - echo srvr |nc localhost 2181

```
[root@localhost bin]# echo stat |nc localhost 2181
Zookeeper version: 3.4.6-1569965, built on 02/20/2014 09:09 GMT
Clients:
 /127.0.0.1:34148[0](queued=0,recved=1,sent=0)
 /0:0:0:0:0:0:0:1:43654[1](queued=0,recved=1,sent=1)
 /0:0:0:0:0:0:0:1:43653[1](queued=0,recved=1,sent=1)

Latency min/avg/max: 0/0/106
Received: 4932
Sent: 4931
Connections: 3
Outstanding: 0
Zxid: 0x9
Mode: standalone
Node count: 6
```

```
Node count: 6
[root@localhost bin]# echo srvr |nc localhost 2181
Zookeeper version: 3.4.6-1569965, built on 02/20/2014 09:09
Latency min/avg/max: 0/0/106
Received: 5037
Sent: 5036
Connections: 3
Outstanding: 0
Zxid: 0x9
Mode: standalone
Node count: 6
[root@localhost bin]#
```

第九讲 四字命令

- srst
  - 重置服务器的统计信息
  - telnet localhost 2181,然后执行srst
  - echo srst |nc localhost 2181

```
Node count: 6
[root@localhost bin]# echo srvr |nc localhost 2181
Zookeeper version: 3.4.6-1569965, built on 02/20/2014 09:09
Latency min/avg/max: 0/0/106
Received: 5037
Sent: 5036
Connections: 3
Outstanding: 0
Zxid: 0x9
Mode: standalone
Node count: 6
[root@localhost bin]#
```

```
[root@localhost bin]# echo srst |nc localhost 2181
Server stats reset.
[root@localhost bin]# echo srvr |nc localhost 2181
Zookeeper version: 3.4.6-1569965, built on 02/20/2014 09:09 GMT
Latency min/avg/max: 0/0/0
Received: 1
Sent: 1
Connections: 3
Outstanding: 0
Zxid: 0x9
Mode: standalone
Node count: 6
[root@localhost bin]#
```

第九讲 四字命令

- wchs
  - 用于输出当前服务器上管理的watcher的概要信息，通过zk构造器创建的默认watcher不在此统计范围
  - telnet localhost 2181,然后执行wchs
  - echo wchs |nc localhost 2181

```
Total watches:1
[root@localhost bin]# echo wchs |nc localhost 2181
1 connections watching 1 paths
Total watches:1
```

第九讲 四字命令

- wchc
  - 用于输出当前服务器上管理的watcher的详细信息，以会话单位为组，通过zk构造器创建的默认watcher不在此统计范围
  - telnet localhost 2181,然后执行wchc
  - echo wchc |nc localhost 2181

```
[root@localhost bin]# echo wchc |nc localhost 2181
0x1522687d7bd000b
        /watchertest
        /test
```

第九讲 四字命令

- wchp
  - 与wchc类似，但是以节点路径分组，通过zk构造器创建的默认watcher不在此统计范围
  - telnet localhost 2181,然后执行wchp
  - echo wchp |nc localhost 2181

```
[root@localhost bin]# echo wchc |nc localhost 2181
0x1522687d7bd000b
        /watchertest
        /test
```

```
[root@localhost bin]# echo wchp |nc localhost 2181
/watchertest
        0x1522687d7bd000b
/test
        0x1522687d7bd000b
```

第九讲 四字命令

- mntr
  - 与stat类似，但是比stat更详细，包括请求的延迟情况 服务区内存数据库大小，集群同步情况等
  - telnet localhost 2181,然后执行mntr
  - echo mntr |nc localhost 2181

```
zk_version  3.4.6-1569965, built on 02/20/2014 09:09 GMT
zk_avg_latency  1
zk_max_latency  1527
zk_min_latency  0
zk_packets_received     5467
zk_packets_sent 5466
zk_num_alive_connections        2
zk_outstanding_requests 0
zk_server_state follower
zk_znode_count  43
zk_watch_count  0
zk_ephemerals_count     1
zk_approximate_data_size        1065
```

```
Zookeeper version: 3.4.6-1569965, built on 02/20/2014 09:09 GMT
Clients:
 /0:0:0:0:0:0:0:1:26094[1](queued=0,recved=5431,sent=5431)
 /0:0:0:0:0:0:0:1:44364[0](queued=0,recved=1,sent=0)

Latency min/avg/max: 0/1/1527
                        Received: 5479
                                Sent: 5478
                                        Connections: 2
                                                Outstanding: 0
                                                        Zxid: 0x1600000010
                                                                Mode: follower
                                                                        Node count: 43
```

第九讲 四字命令

Follower上执行

```
    zk_version  3.4.6-1569965, built on 02/20/2014 09:09 GMT
zk_avg_latency  11
zk_max_latency  80
zk_min_latency  1
zk_packets_received     13
zk_packets_sent 12
zk_num_alive_connections        2
zk_outstanding_requests 0
zk_server_state follower
zk_znode_count  5
zk_watch_count  0
zk_ephemerals_count     0
zk_approximate_data_size        38
```

leader上执行

```
    zk_version  3.4.6-1569965, built on 02/20/2014 09:09
zk_avg_latency  8
zk_max_latency  30
zk_min_latency  1
zk_packets_received     11
zk_packets_sent 10
zk_num_alive_connections        2
zk_outstanding_requests 0
zk_server_state leader
zk_znode_count  5
zk_watch_count  0
zk_ephemerals_count     0
zk_approximate_data_size        38
zk_followers    1
zk_synced_followers     1
zk_pending_syncs        0
```

第九讲 日常运维

- 性能优化
  - Zk本身是基于java实现的，并且数据全量存储在内存中，因此，调大JVM内存是优化点之一，具体数额需要根据业务情况来定，涉及到的JVM参数为-Xmx –Xms -Xmn
  - IO优化:把事务日志与快照存储分磁盘存储，提高IOPS，最好事务日志单独磁盘挂载
  - 加大linux系统的文件句柄数和用户线程数，通过linux的命令ulimit可以查看当前的配置
  - 业务并发高时，可以创建多于1个的客户端会话；可以不同的业务模块采用不同的客户端实例
  - 利用zk进行业务开发时，尽量通过良好的设计减少资源消耗，比如watcher的数量
  - 节点数量，在写少，读多应用场景，采用多一点的节点会提升整体的读并发性能
  - 节点数据最好比默认的10M还小
  - 带宽尽量高，可以通过网络监控查看带宽是否已经是瓶颈

第九讲 日常运维

- 扩容
  - 停机
    - 增加相应的节点即可，比较简单
  - 不停机
    - 增加新的节点，ID一定要比原来集群的要大
    - 新节点启动后会加入集群，并且同步数据
    - 当用mntr命令查看新的节点数据已经同步成功后做下面的操作
    - 按照之前的id顺序依次再去关闭zk实例，然后修改配置，启动实例

知识点纠错说明：
在第五讲课程的第8页中讲到连接是由id小的向大的发起连接，避免重复连接；修正为id大的向小的发起连接，在zk实现中，当发现自己的id比发起连接的id还大时，会关闭此连接

第九讲 日常运维

- 容灾
  - 单机房的容灾靠zk本身的集群机制就能很好的支撑
  - 多机房的容灾
    - 由于过半投票机制，zk不支持双机房的容灾，比如是5节点，分为2和3，当3这个机房出现故障，2不能选举成功
    - 因此，多机房容灾主要是考虑三机房情况
    - 跨机房的网络延时较大，做这个容灾要避免大量写的应用场景
  - 为了避免服务器的地址变化影响客户端，客户端尽量采用域名的方式

第九讲 日常运维

- 重点监控指标
  - Zookeeper事务日志
    - 磁盘IO
    - 可以开启事务日志自动清理
      - autopurge.snapRetainCount
      - autopurge.purgeInterval=24
  - 连接数
  - 注册的watchers数
  - ZK事件通知的延时是否过大

Thanks

FAQ时间