目录

1. 实验一：ZooKeeper 组件部署

1.1. 实验目的

完成本实验，您应该能够：

- 了解 ZooKeeper 的架构
- 了解 ZooKeeper 的集群模式安装

1.2. 实验要求

- 熟悉常用 Linux 操作系统命令
- 熟悉 ZooKeeper 部署

1.3. 实验环境

本实验所需之主要资源环境如表 1-1 所示。

表 1-1 资源环境

服务器集群	3 个以上节点，节点间网络互通，各节点最低配置：双核 CPU、8GB 内存、100G 硬盘
运行环境	CentOS 7.4 (1708) gui 版本
用户名/密码	root/password hadoop/password
服务和组件	HDFS、YARN、MapReduce 等，其他服务根据实验需求安装

1.4. 实验视图

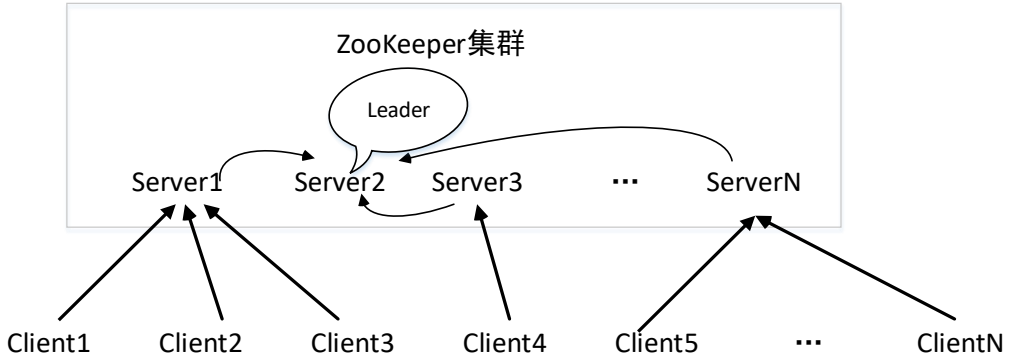

图 1-1 ZooKeeper 架构图

1.5. 实验过程

1.5.1. 实验任务一：集群模式部署 ZooKeeper(第 4 章已安装)

1.5.1.1. 步骤一：解压安装

[root@master ~]# tar -zxvf /opt/software/zookeeper-3.4.8.tar.gz -C /usr/local/src/

[root@master ~]# mv /usr/local/src/zookeeper-3.4.8/ /usr/local/src/zookeeper

1.5.1.2. 步骤二：修改配置文件

Zookeeper 的配置文件放置在 conf 下，提供 zoo_sample.cfg 样例，可重命名 zoo.cfg 后在此基础上修改。

[root@master ~]# cd /usr/local/src/zookeeper/conf/

[root@master ~]# mv zoo_sample.cfg /usr/local/src/zookeeper/conf/zoo.cfg

[root@master ~]# vi /usr/local/src/zookeeper/conf/zoo.cfg

完整配置内容如下：

tickTime=2000

initLimit=10

syncLimit=5

clientPort=2181

dataDir=/usr/local/src/zookeeper/data

dataLogDir=/usr/local/src/zookeeper/logs

server.1=master:2888:3888

server.2=slave1:2888:3888

server.3=slave2:2888:3888

tickTime：心跳时间，单位为毫秒，Leader 服务器与 Follower 服务器连接超时时长。

initLimit：Leader 服务器与 Follower 服务器连接初始化的心跳数。即超过 10 个心跳后 Follower 服务器没有接收到 Leader 服务器，则认为连接失败。与 tickTime 共同使用，每次心跳时间为 2000 毫秒，10*2000=20 秒为连接超时时长。

syncLimit：Leader 服务器与 Follower 服务器相互之间传输信息时能够等待的最大心跳时间，与 tickTime 共同使用。

clientPort：连接 ZooKeeper 的访问端口。

dataDir，dataLogDir：ZooKeeper 的数据存储与日志存储目录。

[root@master ~]# mkdir -p /usr/local/src/zookeeper/data

[root@master ~]# mkdir -p /usr/local/src/zookeeper/logs

除了修改 zoo.cfg 配置文件，集群模式下还要配置一个文件 myid，这个文件在 dataDir 目录下。

[root@master ~]# vi /usr/local/src/zookeeper/data/myid

1

vi 新建文件，并写入数据。以 master 节点为例，文件中只写入一个数字：1。ZooKeeper 读取这个文件，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那个 server。即为 server.1=master:2888:3888 中的 server.X。本实验在 master 节点下配置，则为:1。

1.5.1.3. 步骤三：配置 ZooKeeper 环境变量

[root@master ~]# vi /etc/profile

添加如下配置：

#zookeeper environment

export ZK_HOME=/usr/local/src/zookeeper

export PATH=$PATH:$ZK_HOME/bin 保存并退出

1.5.1.4. 步骤四：文件分发

将 master 主节点已经配置好 ZooKeepr 文件分发给集群从节点。

[root@master ~]# scp -r /usr/local/src/zookeeper root@slave1:/usr/local/src/

[root@master ~]# scp -r /usr/local/src/zookeeper root@slave2:/usr/local/src/
[root@master ~]# scp /etc/profile root@slave1:/etc/
[root@master ~]# scp /etc/profile root@slave2:/etc/

此时需要注意的是，在分发的从节点上需要对每一个 myid 文件进行修改，如 slave1 从节点修改为:2。与 zoo.cfg 配置文件相对应。

[root@ slave1 ~]# vi /usr/local/src/zookeeper/data/myid

2

[root@ slave2 ~]# vi /usr/local/src/zookeeper/data/myid

3

1.5.1.5. 步骤五：修改 ZooKeeper 安装目录的归属用户为 hadoop 用户。

[root@master ~]# chown -R hadoop:hadoop /usr/local/src/zookeeper
[root@slave1 ~] # chown -R hadoop:hadoop /usr/local/src/zookeeper
[root@slave2 ~] # chown -R hadoop:hadoop /usr/local/src/zookeeper

1.5.1.6. 步骤六：启动

关闭防火墙

[root@master ~]# systemctl stop firewalld.service
[root@slave1 ~]# systemctl stop firewalld.service
[root@slave2 ~]# systemctl stop firewalld.service

关闭防火墙自启

[root@master ~]# systemctl disable firewalld.service
Removed symlink /etc/systemd/system/multi-user.target.wants/firewalld.service.
Removed symlink /etc/systemd/system/dbus-org.fedoraproject.FirewallD1.service.
[root@slave1 ~]# systemctl disable firewalld.service
Removed symlink /etc/systemd/system/multi-user.target.wants/firewalld.service.
Removed symlink /etc/systemd/system/dbus-org.fedoraproject.FirewallD1.service.
[root@slave2 ~]# systemctl disable firewalld.service
Removed symlink /etc/systemd/system/multi-user.target.wants/firewalld.service.
Removed symlink /etc/systemd/system/dbus-org.fedoraproject.FirewallD1.service.

[root@master ~]# su - hadoop
[hadoop@master ~]$ source /etc/profile

[hadoop@master ~]$ ./zkServer.sh start

ZooKeeper JMX enabled by default

Using config: /usr/local/src/zookeeper/bin/../conf/zoo.cfg

Starting zookeeper ... STARTED

启动完毕后可通过以下命令查询 Leader 节点与 Follower 节点。本实验经过内部选举 slave1 为 Leader 节点。

[hadoop@master ~]$ ./zkServer.sh status

ZooKeeper JMX enabled by default

Using config: /usr/local/src/zookeeper/bin/../conf/zoo.cfg

Mode: follower

[root@slave1 ~]# su - hadoop

[hadoop@slave1 ~]$ source /etc/profile

[hadoop@slave1 ~]$ ./zkServer.sh start

[hadoop@slave1 ~]$ ./zkServer.sh status

ZooKeeper JMX enabled by default

Using config: /usr/local/src/zookeeper/bin/../conf/zoo.cfg

Mode: follower

[root@slave2 ~]# su - hadoop

[hadoop@slave2 ~]$ source /etc/profile

[hadoop@slave2 ~]$ ./zkServer.sh start

[hadoop@slave2 ~]$ ./zkServer.sh status

ZooKeeper JMX enabled by default

Using config: /usr/local/src/zookeeper/bin/../conf/zoo.cfg

Mode: leader

2. 实验二 ZooKeeper shell 操作

2.1. 实验内容与目标

完成本实验，您应该能够：

- 掌握 ZooKeeper 命令行操作

2.2. 实验视图

图 1-2 ZooKeeper 树结构

根节点是/，Zookeeper 是一个树状结构—Znode 树，每一个 Zookeeper 节点是 Znode 节点。Zookeeper 通过监听树结构下 Clients，维护集群的统一。

2.3. 实验环境

服务器集群	3 个以上节点，节点间网络互通，各节点最低配置：双核 CPU、8GB 内存、100G 硬盘
运行环境	CentOS 7.2
用户名/密码	root/password hadoop/password
服务和组件	HDFS、Yarn、MapReduce

2.4. 实验过程

2.4.1. 实验任务一： ZooKeeper shell 操作

2.4.1.1. 步骤一： ls 命令

连接 zookeeper 服务

[hadoop@master ~]$ zkCli.sh -server localhost:2181

语法：ls [path]

跟 Linux 命令语法一致，在 ZooKeeper 中查看该节点下某一路径下目录列表。

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

[zookeeper,yarn-leader-election,hadoop-ha,hbase,rmstore]

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

[quota]

[zk: localhost:2181(CONNECTED) 2] ls /zookeeper/quota

[]

2.4.1.2. 步骤二： stat 命令

语法：stat [path]
显示节点的详细信息。

[zk: localhost:2181(CONNECTED) 3] stat /

cZxid = 0x0

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

mZxid = 0x0

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

pZxid = 0x300000075

cversion = 3

dataVersion = 0

aclVersion = 0

ephemeralOwner = 0x0

dataLength = 0

numChildren = 5

cZxid:代表 zookeeper 创建后为这个节点所分配的 ID 号。

ctime:代表该节点创建时间。

mZxid:代表修改节点后分配的一个新 ID。节点每次修改 mZxid 都会以递增的形式增加。

mtime:代表最后一次被修改的时间。

pZxid:代表节点的子节点的最后被修改生成的 ID 号。

cversion:节点下所有子节点被修改的版本号，表示节点被修改的次数。

dataVersion:节点修改的版本号，代表节点被修改的次数，初始值为 0，每一次修改一次值加 1。

aclVersion:代表一个权限模型 acl 代表权限 当节点权限发生变化的时候 权限的版本会自动累加 1

ephemeralOwner：如果节点为临时节点，表示节点拥有者的会话 ID，否则为 0.

dataLength:代表的是元数据长度

numChildren:节点下子节点的数量。

2.4.1.3. 步骤三： ls2 命令

语法：ls2 [path]
相当于 ls + stat。显示当前目录与详细信息。

```
[zk: master:2181(CONNECTED) 5] ls2 /
[zookeeper, yarn-leader-election, hadoop-ha, hbase, rmstore]
cZxid = 0x0
ctime = Thu Jan 01 08:00:00 CST 1970
mZxid = 0x0
mtime = Thu Jan 01 08:00:00 CST 1970
pZxid = 0x300000075
cversion = 3
dataVersion = 0
aclVersion = 0
ephemeralOwner = 0x0
dataLength = 0
numChildren = 5
```

2.4.1.4. 步骤四：create 创建节点

语法：create [-s] [-e] [path]<node_name> <data>

创建节点，本实验在根目录下创建 zknode1 节点，元数据为字符串 node1_data。也可以在节点下创建子节点。[-s]表示顺序创建，[-e]表示创建的是临时节点。

```
[zk: master:2181(CONNECTED) 6] create /zknode1 "node1_data"
Created /zknode1
[zk: master:2181(CONNECTED) 7] create /zknode1/node1 "data"
Created /zknode1/node1
```

2.4.1.5. 步骤五：get 命令

语法：get [path]<node_name>

获取当前的目录（节点）的数据信息。

```
[zk: master:2181(CONNECTED) 9] get /zknode1
node1_data
cZxid = 0x100000003
ctime = Wed Jul 05 05:15:43 CST 2017
mZxid = 0x100000003
mtime = Wed Jul 05 05:15:43 CST 2017
pZxid = 0x100000004
cversion = 1
dataVersion = 0
aclVersion = 0
ephemeralOwner = 0x0
dataLength = 10
numChildren = 1
```

2.4.1.6. 步骤六：set 命令

语法：set [path]<node_name> <data>

修改节点的元数据。本实验元数据修改为"zknode1_data"，get 查看后 dataVersion 加 1，数据的长度也发生了变化。

```
[zk: master:2181(CONNECTED) 10] set /zknode1 "zknode1_data"
cZxid = 0x100000003
```

ctime = Wed Jul 05 05:15:43 CST 2017
mZxid = 0x100000005
mtime = Wed Jul 05 05:37:09 CST 2017
pZxid = 0x100000004
cversion = 1
dataVersion = 1
aclVersion = 0
ephemeralOwner = 0x0
dataLength = 12
numChildren = 1
[zk: master:2181(CONNECTED) 11] get /zknode1
zknode1_data
cZxid = 0x100000003
ctime = Wed Jul 05 05:15:43 CST 2017
mZxid = 0x100000005
mtime = Wed Jul 05 05:37:09 CST 2017
pZxid = 0x100000004
cversion = 1
dataVersion = 1
aclVersion = 0
ephemeralOwner = 0x0
dataLength = 12
numChildren = 1

2.4.1.7. 步骤七：删除节点

语法：delete [path]<node_name>
删除节点要求该节点下没有子节点。
[zk: master:2181(CONNECTED) 12] delete /zknode1
Node not empty: /zknode1
[zk: master:2181(CONNECTED) 13] delete /zknode1/node1
[zk: master:2181(CONNECTED) 14] delete /zknode1