Zookeeper 学习总结

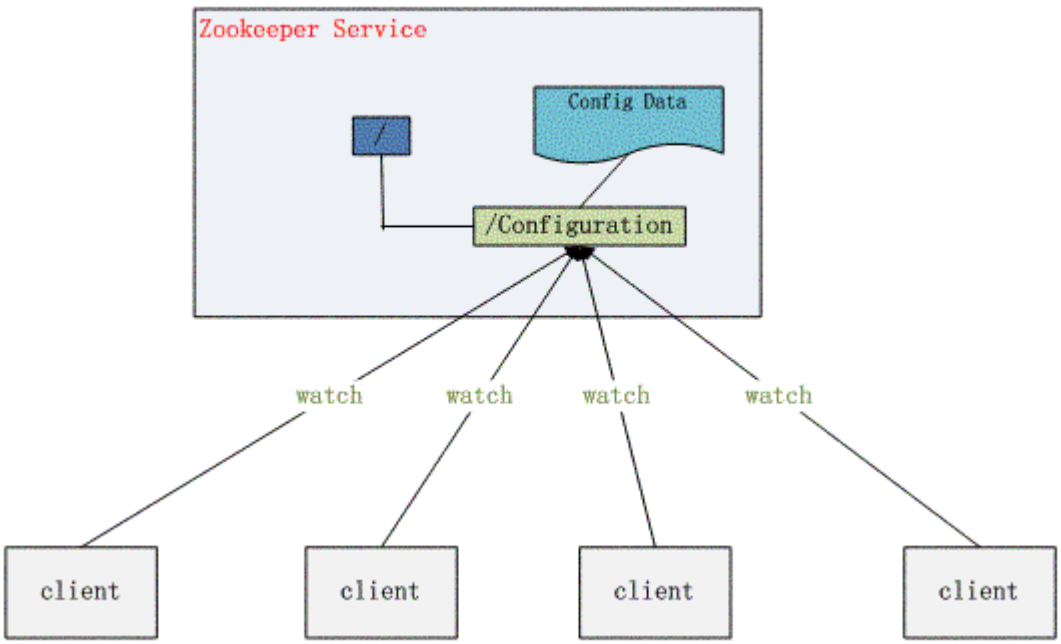
——以 hadoop HA 集群环境搭建为主

Zookeeper 的官方定义是：ZooKeeper is a centralized service for maintaining configuration information, naming, providing distributed synchronization, and providing group services.

总而言之，Zookeeper 是一个高性能、分布式、开源分布式应用协调服务，可以提供下列功能:配置信息管理、命名管理、提供分布式同步、集群管理。

1. 配置信息管理

   如果只有少数机器，也不需要经常修改配置文件，那么在服务器上使用配置文件不会造成不方便。但是如果程序分散在多台机器上，或者配置文件需要经常进行修改是，那么在服务器上使用配置文件就不是一个好的方法，此时就可以使用 Zookeeper 将配置保存在 Zookeeper 的节点中，然后应用程序对目录节点进行监听，一旦配置信息变化，就会收到 Zookeeper 的通知，并且把配置信息更新就可以。



2. 命名管理

   在分布式系统中，通过使用 Zookeeper 命名服务，客户端的应用可以根据指定的名字获取资源和服务的地址等信息。在 Zookeeper 的文件系统里创建一个目录，即有唯一的 path。在我们使用 tborg 无法确定上游程序的部署机器时，即可与下游程序约定好 path，通过 path 即能互相发现。

3. 分布式锁

   Zookeeper 的锁服务可以分为两类，分别是保持独占和控制时序。

   对于保持独占，我们将 Zookeeper 上的 znode 看成是一把锁，通过 createznode 的方式来实现。所有客户端创建/distribute_lock 节点，成功创建的客户端就有了这把锁，用完删掉该节点就释放锁。

   对于控制时序，/distribute_lock 已经预先存在，所有客户端在下面创建临时顺序编号目

录节点，和选择 master 一样，编号最小的获得锁，依次用完，然后删除。

4. 集群管理

   集群管理在于：是否有机器加入和退出，选举 master

   在分布式集群中，经常会由于硬件软件故障、网络问题等各种原因，会导致节点退出和新的节点加入。这个时候集群中的其他节点就需要及时感知到这些变化并且及时做出反应。在 Zookeeper 中，所有机器约定在父目录 GroupMembers 下创建临时目录节点，然后监听父目录节点的子节点变化消息。一旦有机器挂掉，该机器与 zookeeper 的连接断开，其所创建的临时目录节点被删除，所有其他机器都收到通知。新机器加入也是类似，所有机器收到通知：新兄弟目录加入，highcount 又有了。

   对于选举 master，所有机器创建临时顺序编号目录节点，每次选取编号最小的机器作为 master。

ZooKeeper 的数据模型：

1.树形结构、每个 znode 节点可以存储数据、并有一个关联的 ACL。但 znode 存储的数据被限制在 1MB 内。
2.数据访问具有原子性，要么成功，要么全部失败。
3.znode 通过路径被引用。类似于 linux 的文件路径。
4.znode 有两种，短暂性的和持久性的。短暂性的会在客户端结束后删除
5.顺序号：命名时，zookeeper 在名字中指定了顺序号，这个在创建 znode 时，设置顺序标识就会产生，顺序号，可用于全局排序
6.观察：是一种机制，znode 发生变化时，观察机制可以通知客户端（类似于观察者模式），观察注册后只触发一次
一般在读操作 exists、getChildren 和 getData 上设置观察，然后这些观察在 create、delete、setData 时触发。
7.ACL
每个 znode 被创建都会带有一个 ACL 列表，用于检验权限：谁，可以对它自己执行，何种操作
ACL 依赖于 ZooKeeper 的身份依赖机制。
digest：利用用户名和密码来识别客户端
host：通过主机名 hostname 来识别客户端
ip：利用 ip 来识别客户端

ZooKeeper 的实现方式：

独立模式：适用于测试环境，甚至单元测试。只有一个 ZooKeeper 服务器，但是不保证高可用性和恢复性。
复制模式：适用于生产环境，通过复制来保证高可用性和恢复性，保证半数以上的机器处于可用状态。
其实现使用 Zab 协议，包括两个阶段：
阶段一：领导者选举： 一台机器选为 leader，其他为 follower，一旦半数以上的机器与 leader 的同步状态相同，此阶段完成。
阶段二：原子广播：所有写请求给 leader，然后再由 leader 广播给 follower，且当半数以上 follower 完成同步后，leader 才提交这个更新，然后客户端才收到完成的响应。

若领导者故障，则由 follower 中选出新的 leader。
以上两个过程可以无限循环执行。

Zookeeper 的特征：

其核心是一个精简的文件系统；
其原语是一组丰富的 “构件”，可用于实现很多数据结构和协议，如分布式队列，分布式锁，同一级中的领导者选举；
能避免单点故障，高可用性；
松耦合交互方式：各进程间不必相互了解，同步等；
是一个资源库，对通用协议提供一个开源的共享存储库。

下面选取三台机器建立 hadoop HA 集群，并通过 Zookeeper 管理

	hadoop101 10.131.226.199	hadoop102 10.131.226.208	hadoop103 10.131.226.215
是 NameNode 吗?	是，属集群 c1	是，属集群 c1	不是，属集群 c1
是 DataNode 吗？	不是	是	是
是 JournalNode 吗？	是	是	不是
是 ZooKeeper 吗？	是	是	是
是 ZKFC 吗?	是	是	是

修改每台机器的/etc/hosts，增加
10.131.226.199 hadoop101
10.131.226.208 hadoop102
10.131.226.215 hadoop103

配置 zookeeper 集群
在 hadoop101、hadoop102、hadoop103 上都安装 zookeeper
根据 Zookeeper 集群节点情况，在 conf 目录下创建 Zookeeper 配置文件 zoo.cfg：

```
tickTime=2000
dataDir=/var/zookeeper/
clientPort=2181
initLimit=5
syncLimit=2
server.1=hadoop101:2888:3888
server.2=hadoop102:2888:3888
server.3=hadoop103:2888:3888
```

其中，dataDir 指定 Zookeeper 的数据文件目录；其中 server.id=host:port:port，id 是为每个 Zookeeper 节点的编号，保存在 dataDir 目录下的 myid 文件中，zoo1~zoo3 表示各个 Zookeeper 节点的 hostname，第一个 port 是用于连接 leader 的端口，第二个 port 是用于 leader 选举的端口。

启动 Zookeeper 服务：

```
bin/zkServer.sh start
```

接着进行文件配置：
配置文件一共包括 6 个，分别是 hadoop-env.sh、core-site.xml、hdfs-site.xml、mapred-site.xml、yarn-site.xml 和 slaves，在三个节点上都是一样的，可以使用 SSH 无密码链接传输。

配置启动完成后，执行$ZOOKEEPER_HOME/bin/zkCli.sh 查看节点，效果如下：

```
WatchedEvent state:SyncConnected type:None path:null
[zk: localhost:2181(CONNECTED) 0] ls /
[hadoop-ha, zookeeper]
[zk: localhost:2181(CONNECTED) 1] ls /hadoop-ha
[cluster1]
[zk: localhost:2181(CONNECTED) 2]
```

接着启动 JournalNode 集群，并且格式化集群 c1 的一个 NameNode
启动刚才格式化的 NameNode，可以看到

```
[root@hadoop101 hadoop-2.6.4]# sbin/hadoop-daemon.sh start namenode
starting namenode, logging to /root/hadoop/hadoop-2.6.4/logs/hadoop-root-namenode-hadoop101.out
[root@hadoop101 hadoop-2.6.4]# jps
2823 Jps
2752 NameNode
2209 QuorumPeerMain
2528 JournalNode
[root@hadoop101 hadoop-2.6.4]#
```

启动后，产生一个新的 java 进程 NameNode

把 NameNode 的数据从 hadoop101 同步到 hadoop102 中，在 hadoop102 上启动 c1 中另一个 Namenode

```
[root@hadoop102 hadoop-2.6.4]# sbin/hadoop-daemon.sh start namenode
starting namenode, logging to /root/hadoop/hadoop-2.6.4/logs/hadoop-root-namenode-hadoop102.out
[root@hadoop102 hadoop-2.6.4]# jps
2185 QuorumPeerMain
2676 Jps
2344 JournalNode
2605 NameNode
[root@hadoop102 hadoop-2.6.4]#
```

启动所有的 DataNode

```
[root@hadoop101 hadoop-2.6.4]# sbin/hadoop-daemons.sh start datanode
hadoop102: starting datanode, logging to /root/hadoop/hadoop-2.6.4/logs/hadoop-root-datanode-hadoop102.out
hadoop103: starting datanode, logging to /root/hadoop/hadoop-2.6.4/logs/hadoop-root-datanode-hadoop103.out
[root@hadoop101 hadoop-2.6.4]#
```

上文的配置中 hadoop101 不是 datanode，hadoop102 跟 hadoop103 都是 datanode

hadoop102 上的 jps：

```
[root@hadoop102 hadoop-2.6.4]# jps
2185 QuorumPeerMain
3067 Jps
2344 JournalNode
2981 DataNode
2605 NameNode
[root@hadoop102 hadoop-2.6.4]#
```

hadoop103 上的 jps：

```
[root@hadoop103 hadoop]# jps
2186 QuorumPeerMain
2503 DataNode
2600 Jps
[root@hadoop103 hadoop]#
```

启动 yarn，启动 ZooKeeperFailoverController

```
[root@hadoop101 hadoop-2.6.4]# sbin/hadoop-daemon.sh start zkfc
starting zkfc, logging to /root/hadoop/hadoop-2.6.4/logs/hadoop-root-zkfc-hadoop101.out
[root@hadoop101 hadoop-2.6.4]# jps
3475 Jps
2752 NameNode
2209 QuorumPeerMain
3090 ResourceManager
3412 DFSZKFailoverController
2528 JournalNode
[root@hadoop101 hadoop-2.6.4]#
```

有了 ZooKeeperFailoverController，namenode 就能起到 ha 的作用了。

对于 HA 集群而言，确保同一时刻只有一个 NameNode 处于 active 状态是至关重要的。否则，两个 NameNode 的数据状态就会产生分歧，可能丢失数据，或者产生错误的结果。为了保证这点，这就需要利用使用 ZooKeeper 了。首先 HDFS 集群中的两个 NameNode 都在 ZooKeeper 中注册，当 active 状态的 NameNode 出故障时，ZooKeeper 能检测到这种情况，它就会自动把 standby 状态的 NameNode 切换为 active 状态。

由于大多数都是在命令行里执行的，或者是配置文件，所以没有提供代码。