大数据技术基础

实验三 HBase、Zookeeper 及 HBase 应用实践

鄂海红 计算机学院（国家级示范性软件学院） 教授

ehaihong@bupt.edu.cn 微信: 87837001 QQ: 3027581960

HBase实验相关知识点

北京郵電大學
BEIJING UNIVERSITY OF POSTS AND TELECOMMUNICATIONS

1 Zookeeper基本知识

2 Zookeeper在HBase的应用

3 Zookeeper和HBase安装要点

ZooKeeper

- ZooKeeper是一个开源的分布式协调服务，由雅虎创建，是Google Chubby的开源实现
- 分布式应用程序可以基于ZooKeeper实现诸如数据发布/订阅、负载均衡、命名服务、分布式协调/通知、集群管理、Master选举、分布式锁和分布式队列等功能
- 在ZooKeeper中，有三种角色：Leader、Follower、Observer
- 一个ZooKeeper集群同一时刻只会有一个Leader，其他都是Follower或Observer

ZooKeeper配置

- 每个节点的配置文件(zoo.cfg)都是一样的，只有myid文件不一样。myid的值必须是zoo.cfg中server.{数值}的{数值}部分。

zoo.cfg配置

```
maxClientCnxns=0
# The number of milliseconds of each tick
tickTime=2000
# The number of ticks that the initial
# synchronization phase can take
initLimit=10
# The number of ticks that can pass between
# sending a request and getting an acknowledgement
syncLimit=5
# the directory where the snapshot is stored.
dataDir=/var/lib/zookeeper/data
# the port at which the clients will connect
clientPort=2181
# the directory where the transaction logs are stored.
dataLogDir=/var/lib/zookeeper/logs
server.1=192.168.20.101:2888:3888
server.2=192.168.20.102:2888:3888
server.3=192.168.20.103:2888:3888
server.4=192.168.20.104:2888:3888
minSessionTimeout=4000
maxSessionTimeout=100000
```

myid文件创建、写入数值

创建tmp目录作数据目录：

```
mkdir /usr/local/zookeeper/tmp
```

在tmp目录中创建一个空文件myid，并向该文件写入ID：

```
touch /usr/local/zookeeper/tmp/myid
echo 1 > /usr/local/zookeeper/tmp/myid
```

name-number-0002：

```
cd /usr/local
ln -s zookeeper-3.4.6 zookeeper
echo 2 > /usr/local/zookeeper/tmp/myid
```

name-number-0003：

```
cd /usr/local
ln -s zookeeper-3.4.6 zookeeper
echo 3 > /usr/local/zookeeper/tmp/myid
```

name-number-0004：

```
cd /usr/local
ln -s zookeeper-3.4.6 zookeeper
echo 4 > /usr/local/zookeeper/tmp/myid
```

ZooKeeper各节点的角色

- 在装有ZooKeeper的机器的终端执行 zookeeper-server status 可以看当前节点的ZooKeeper是什么角色（Leader or Follower）；这是由Zookeeper启动后，自己进行的选举产生的。

```
[root@node-20-103 ~]# zookeeper-server status
JMX enabled by default
Using config: /etc/zookeeper/conf/zoo.cfg
Mode: follower
```

```
[root@node-20-104 ~]# zookeeper-server status
JMX enabled by default
Using config: /etc/zookeeper/conf/zoo.cfg
Mode: leader
```

- ZooKeeper集群的所有机器通过一个Leader选举过程来选定一台被称为『Leader』的机器，Leader服务器为客户端提供读和写服务。
- Follower和Observer都能提供读服务，不能提供写服务。两者唯一的区别在于，Observer机器不参与Leader选举过程，也不参与写操作的『过半写成功』策略，因此Observer可以在不影响写性能的情况下提升集群的读性能。

- ZooKeeper启动后，默认只有Leader和Follower两种角色，没有Observer角色。
- 为了使用Observer模式，在任何想变成Observer的节点的配置文件中加入：peerType=observer
- 并在所有server的配置文件中，配置成observer模式的server的那行配置追加:observer
- 例如：

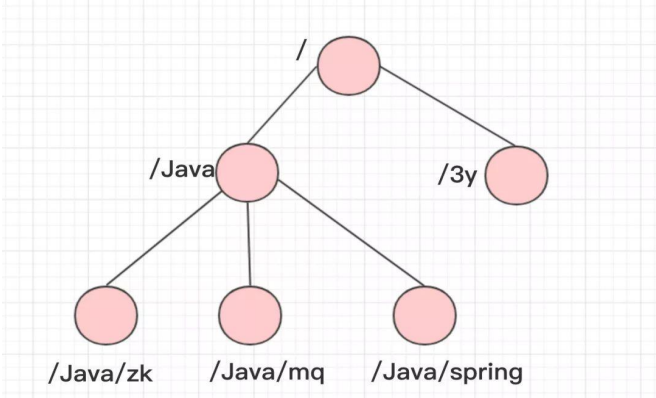
server.1:localhost:2888:3888:observer

zoo.cfg配置

```
maxClientCnxns=0
# The number of milliseconds of each tick
tickTime=2000
# The number of ticks that the initial
# synchronization phase can take
initLimit=10
# The number of ticks that can pass between
# sending a request and getting an acknowledgemen
syncLimit=5
# the directory where the snapshot is stored.
dataDir=/var/lib/zookeeper/data
# the port at which the clients will connect
clientPort=2181
# the directory where the transaction logs are stored.
dataLogDir=/var/lib/zookeeper/logs
server.1=192.168.20.101:2888:3888
server.2=192.168.20.102:2888:3888
server.3=192.168.20.103:2888:3888
server.4=192.168.20.104:2888:3888
minSessionTimeout=4000
maxSessionTimeout=100000
```

ZooKeeper的数据单元ZNode

- ZooKeeper中的ZNode是数据模型中的数据单元，称为在谈到分布式的时候，一般『节点』指的是组成集群的每一台机器，注意这里不是。ZooKeeper将所有数据存储在内存中，数据模型是一棵树（ZNode Tree），由斜杠（/）进行分割的路径，就是一个ZNode，如/hbase/master,其中hbase和master都是ZNode。每个ZNode上都会保存自己的数据内容，同时会保存一系列属性信息。
- 这里的ZNode可以理解成既是Unix里的文件，又是Unix里的目录。因为每个ZNode不仅本身可以写数据（相当于Unix里的文件），还可以有下一级文件或目录（相当于Unix里的目录）。
- 在ZooKeeper中，ZNode可以分为持久节点和临时节点两类。
- 持久节点：所谓持久节点是指一旦这个ZNode被创建了，除非主动进行ZNode的移除操作，否则这个ZNode将一直保存在ZooKeeper上。
- 临时节点：临时节点的生命周期跟客户端会话绑定，一旦客户端会话失效，那么这个客户端创建的所有临时节点都会被移除。

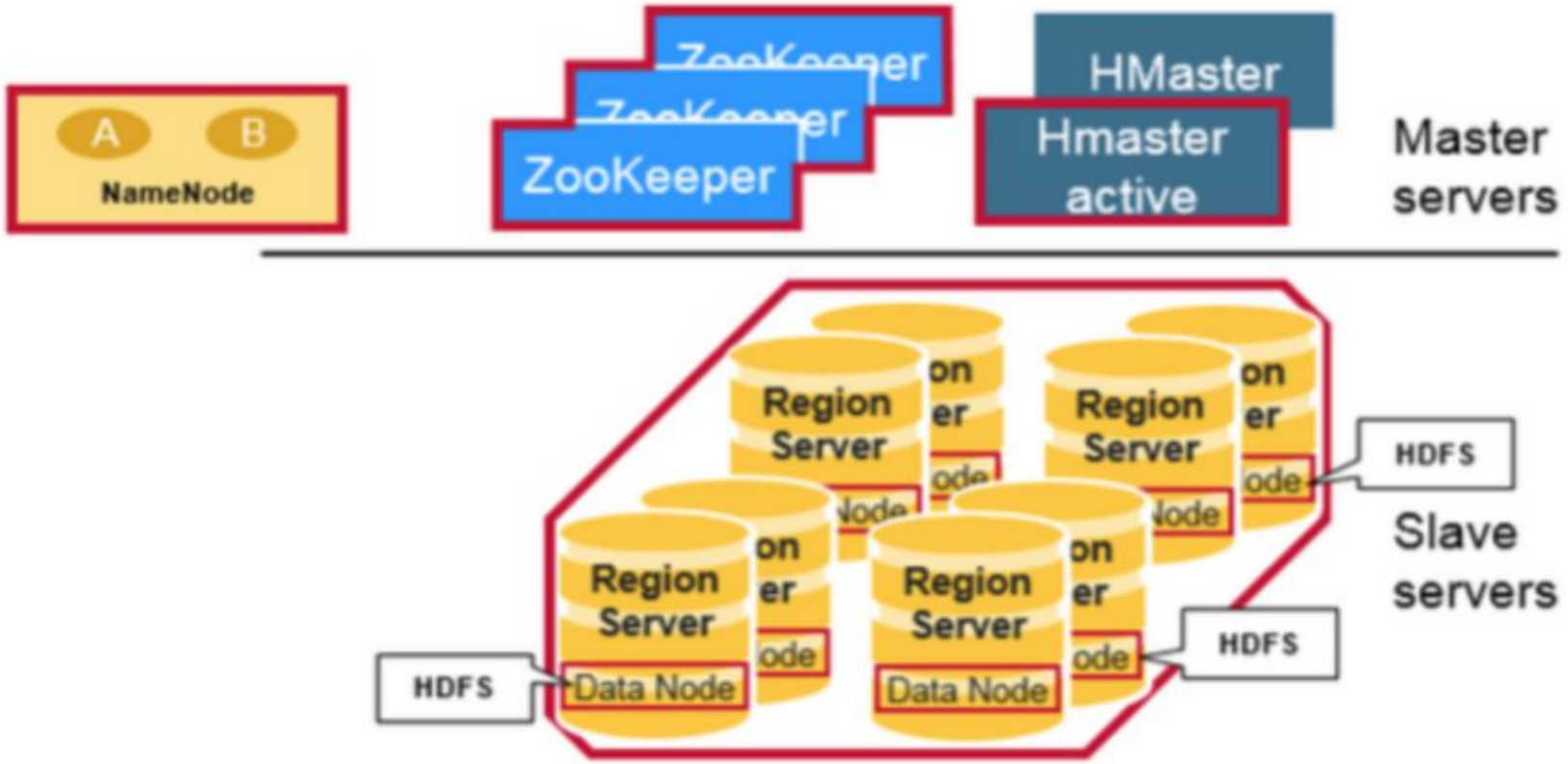


ZooKeeper典型应用场景

- 配置中心
  - 系统中需要使用一些通用的配置信息，例如机器列表信息、数据库配置信息等。这些全局配置信息通常具备以下3个特性：数据量通常比较小；数据内容在运行时动态变化；集群中各机器共享，配置一致。
  - 全局配置信息就可以发布到ZooKeeper上，让客户端（集群的机器）去订阅该消息。
- 命名服务(Naming Service)
  - 在分布式系统中，通过使用命名服务，客户端应用能够根据指定名字来获取资源或服务的地址，提供者等信息。被命名的实体通常可以是集群中的机器，提供的服务，远程对象等等
  - 在ZooKeepr里创建顺序节点，能够很容易创建一个全局唯一的路径，这个路径就可以作为一个名字。ZooKeeper的命名服务即生成全局唯一的ID。
- 分布式协调/通知
  - ZooKeeper中特有Watcher注册与异步通知机制，能够很好的实现分布式环境下不同机器，甚至不同系统之间的通知与协调，从而实现对数据变更的实时处理。使用方法通常是不同的客户端都对ZK上同一个ZNode进行注册，监听ZNode的变化（包括ZNode本身内容及子节点的），如果ZNode发生了变化，那么所有订阅的客户端都能够接收到相应的Watcher通知，并做出相应的处理。
- Master选举
  - Master选举可以说是ZooKeeper最典型的应用场景了
  - 比如HDFS中Active NameNode的选举、YARN中Active ResourceManager的选举和HBase中Active HMaster的选举等。

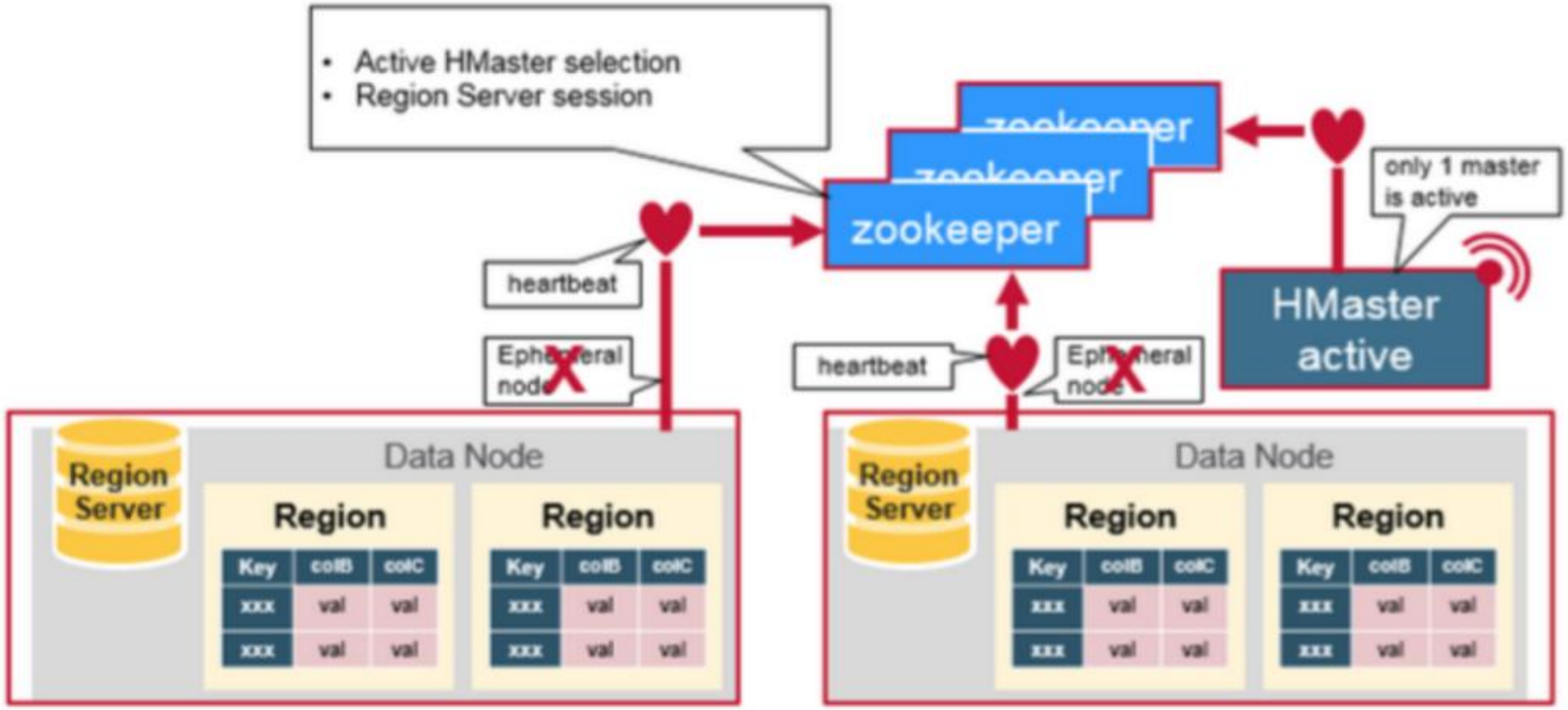
ZooKeeper在HBase集群中的作用

- 当HBase集群启动成功后，会在ZK注册如下znode：
- /hbase/master，其中包含当前活动（即赢得选举）的HMaster信息；
- /hbase/backup-masters/[host-name]，每个子znode包含当前作为热备的HMaster信息；
- /hbase/rs/[host-name]，每个子znode包含各RegionServer的信息。
- 所有znode都是临时（ephemeral）节点，HMaster和RegionServer通过心跳维护这些znode。



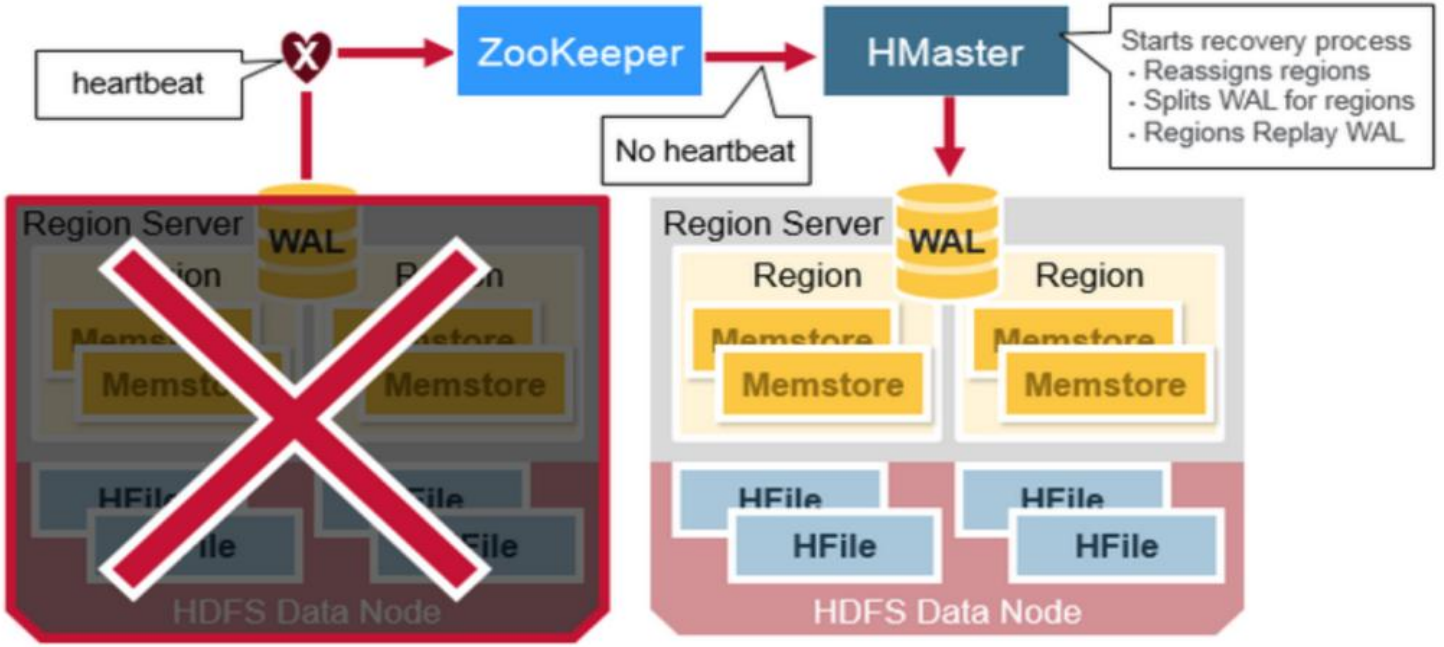
ZooKeeper在HBase集群中的作用

- 活动HMaster对/hbase/rs路径下的znode注册监听，当有RegionServer失败时，心跳信号消失，超时过后其对应的znode被删除，HMaster即可感知到RegionServer下线，并将该RegionServer持有的Region重新路由到其他服务器上去。
- 同理，所有热备HMaster都对/hbase/master节点注册监听，当前HMaster挂掉后，该znode被删除，即可触发重新选举HMaster。



ZooKeeper在HBase集群中的作用

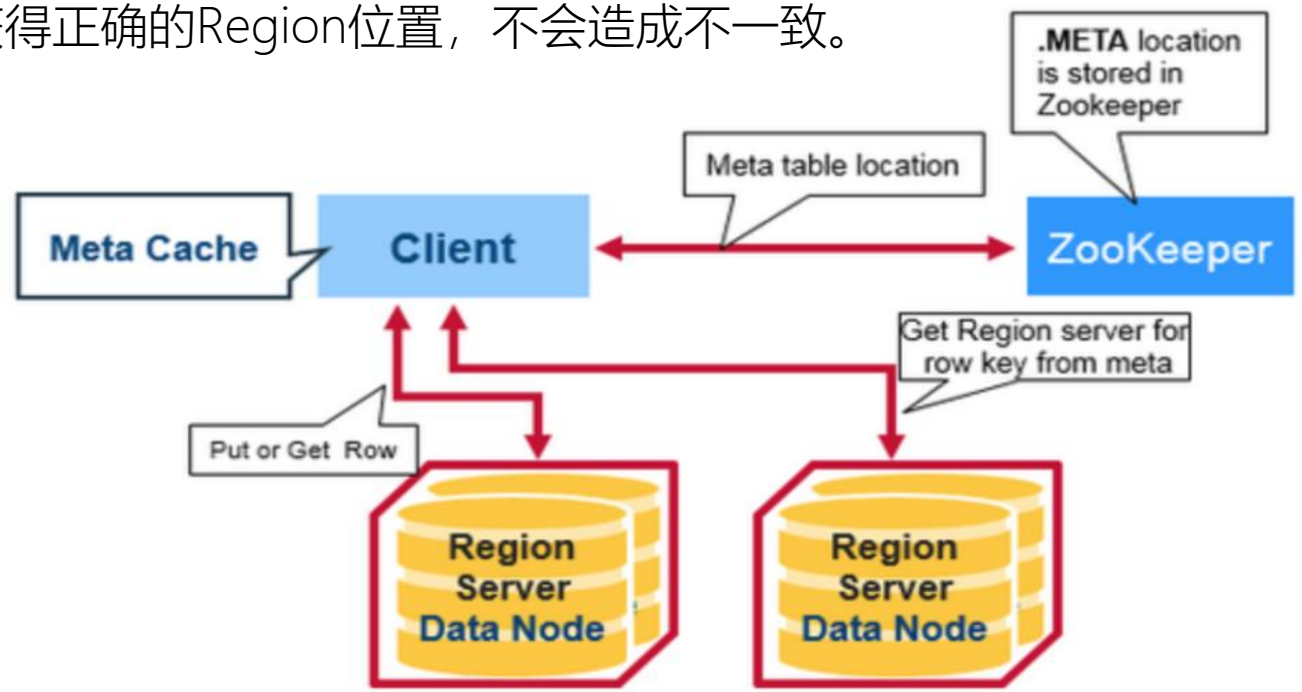
- 当RegionServer宕机时，除了重新路由Region之外，还得从宕机的RegionServer的WAL（即HLog）中恢复尚未持久化到HFile的数据。为了保证尽快完成failover过程，HBase会将HLog按Region切分成多个分片，并分配给对应的存活RegionServer再完成重放（replay）过程。如下图所示。
- HMaster会在ZK上注册/hbase/splitlog临时节点，其中存放有存活RegionServer与其应该处理的Region HLog的映射关系。各个RegionServer从该节点得到分配的Region，重放HLog，并将结果写回该节点，以通知HMaster进行后续操作。



ZooKeeper在HBase集群中的作用

.META.表位置维护

- ZK通过永久（persistent）节点/hbase/meta-region-server来记录.META.表保存在哪个RegionServer上。
- 当客户端初次与HBase集群建立连接时，它首先查询上述ZK节点，再从持有.META.表的RegionServer获取到RowKey对应的Region位置信息并缓存起来，最后获取到对应的行做读写操作。
- 如果Region被移动，或客户端缓存失效，甚至.META.表所在的服务器故障，客户端总能通过ZK维护的路径获得正确的Region位置，不会造成不一致。



实验三介绍

北京郵電大學 BEIJING UNIVERSITY OF POSTS AND TELECOMMUNICATIONS

实验描述：

在实验一、二搭建好的集群环境上，继续安装 HBase、Zookeeper，实践 HBase 基本使用。

实验目的：

掌握 HBase、ZooKeeper 的安装与使用，使用 MapReduce 批量将 HBase 表上的数据导入到 HDFS 中，学习本实验能快速掌握 HBase 数据库在分布式计算中的应用，理解 Java API 读取 HBase 数据等相关内容。

实验手册为HTML形式，解压实验三.zip，点击index.html进入实验手册。

zookeeper 安装

在用户目录下下载 zookeeper 压缩包并解压：

```
wget https://archive.apache.org/dist/zookeeper/zookeeper-3.4.6/zookeeper-3.4.6.tar.gz
```

```
cd /usr/local
```

```
tar -zxvf zookeeper-3.4.6.tar.gz
```

建立软链接，便于后期版本更换：

```
ln -s zookeeper-3.4.6 zookeeper
```

打开配置文件：

```
vim /etc/profile
```

添加 ZooKeeper 到环境变量：

```
export ZOOKEEPER_HOME=/usr/local/zookeeper
```

```
export PATH=$ZOOKEEPER_HOME/bin:$PATH
```

使环境变量生效：

```
source /etc/profile
```

进入 ZooKeeper 所在目录：

```
cd /usr/local/zookeeper/conf
```

拷贝配置文件：

```
cp zoo_sample.cfg zoo.cfg
```

修改配置文件：

```
vim zoo.cfg
```

实验手册中蓝色背景的命令是在终端下执行的。

实验手册中棕黄色背景的命令是要添加到配置文件里的。

zookeeper 安装

修改数据目录：

```
dataDir=/usr/local/zookeeper/tmp
```

在最后添加如下代码，server.1-4 是部署 ZooKeeper 的节点，1，2，3，4 分别是各服务器/usr/local/zookeeper/tmp/myid文件的内容。这里 192.168.0.xxx 对应的是运行 QuorumPeerMain 的服务器的内网 IP，需要改成自己集群的。

```
server.1=192.168.0.132:2888:3888
server.2=192.168.0.83:2888:3888
server.3=192.168.0.62:2888:3888
server.4=192.168.0.154:2888:3888
```

结点1的IP： 结点2的IP： 结点3的IP： 结点4的IP： 生成

为了防止配置文件修改错误，实验手册提供了配置自动生成功能，输入4个结点的IP，点击“生成”，会自动生成相应的配置，直接复制即可使用。

zookeeper 安装

1. 接下来为Zookeeper创建tmp目录，并将Zookeeper从主节点复制到其它结点，具体操作可参考实验手册。
2. 然后在4个结点上分别启动Zookeeper：

```
cd /usr/local/zookeeper/bin
```

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

如果有如下结果则说明启动成功：

```
[root@name-number-0002 bin]# ./zkServer.sh start
JMX enabled by default
Using config: /usr/local/zookeeper/bin/../conf/zoo.cfg
Starting zookeeper ... STARTED
```

HBase 安装

1. 具体安装过程可以参考实验手册，完成安装后运行如下命令启动Hbase集群：

```
/usr/local/hbase/bin/start-hbase.sh
```

2. 运行jps命令观察进程是否都正常启动，如果主从节点的进程分别如下图所示则安装成功：

```
[root@smz-2022010271-0001 ~]# jps
3054 NameNode
7935 Jps
3273 SecondaryNameNode
3741 QuorumPeerMain
3457 ResourceManager
7736 HMaster
```

主节点进程

```
[root@smz-2022010271-0002 ~]# jps
2288 DataNode
5566 Jps
2561 QuorumPeerMain
5363 HRegionServer
2408 NodeManager
```

从节点进程

HBase 安装

1. 如果安装过程没有遇到问题则基于Hbase完成代码实战。
2. 如果遇到问题可以参考如下方法重启系统：

大多数非配置问题可以通过重启+清空缓存来解决： 在主结点上关闭Hadoop：

```
stop-all.sh
```

对于所有的4台服务器，重新启动：

```
reboot
```

对于所有的4台服务器，清空ZooKeeper的缓存：

```
rm -rf /usr/local/zookeeper/tmp/zookeeper_server.pid
```

```
rm -rf /usr/local/zookeeper/tmp/version-2
```

在主结点上启动Hadoop：

```
start-all.sh
```

对于所有的4台服务器，启动ZooKeeper：

```
/usr/local/zookeeper/bin/zkServer.sh start
```

在主结点上启动Hbase：

```
/usr/local/hbase/bin/start-hbase.sh
```