第5章

HBase 简介、部署与开发

本章首先介绍 HBase 数据库，包括数据模型、系统架构及存储格式。其次介绍 HBase 的安装和配置，以及如何使用 HBase Shell 建表和增加、删除、修改、查询数据。最后剖析了一个 HBase Java 实例。

5.1 HBase 简 介

HBase 是一个高可靠、高性能、面向列、可伸缩的分布式数据库系统。利用 HBase 可在廉价 PC Server 上搭建大规模结构化数据管理集群。

HBase 是一款借鉴 Google Bigtable 技术实现的开源软件。Bigtable 利用 GFS 作为其文件存储系统，HBase 则利用 HDFS 作为其文件存储系统；Bigtable 利用 MapReduce 处理 Bigtable 中的海量数据，HBase 则利用 Hadoop MapReduce 处理 HBase 中的海量数据；Bigtable 利用 Chubby 实现协同服务，HBase 则利用 Zookeeper 实现同样的功能。

在 Hadoop 生态系统中，HBase 为一款结构化数据管理工具。HDFS 为 HBase 提供了高可靠性的底层存储支持，Hadoop MapReduce 为 HBase 提供了高性能的计算能力，Zookeeper 为 HBase 提供了稳定服务和故障转移(Failover)机制。Sqoop 则为 HBase 提供了方便的 RDBMS 数据导入功能，使得从传统数据库向 HBase 迁移变得非常方便。

5.2 HBase 访问接口

HBase 提供了丰富的访问接口。

(1) Native Java API：是最常用且高效的访问方式，适合 MapReduce Job 并行处理 HBase 表格的数据。

(2) HBase Shell：是 HBase 的命令行工具，适合对 HBase 进行管理时使用。

(3) Thrift Gateway：利用 Thrift 序列化技术，支持 C++、PHP、Python 等多种语言，适合其他异构系统访问 HBase 表格数据。

(4) REST Gateway：支持 REST 风格的 HTTP API 访问 HBase，解除了

语言限制。

(5) Pig：可以使用 Pig Latin 编程语言来操作 HBase 中的数据(编译成 MapReduce Job 来处理 HBase 表格的数据)，适合做数据统计。

(6) Hive：0.7.0 以上版本的 Hive 支持 HBase，使得用户可以使用类似 SQL 的查询语言来访问 HBase。

5.3 HBase 的数据模型

HBase 包含如下 4 个重要概念。

Table：HBase 的表格，类似关系数据库的表格，但是有所不同。

Row Key：行键，是 Table 的主键，Table 中的记录按照 Row Key 排序。

Column 和 Column Family(列簇或者列分组)：Table 在垂直方向由一个或者多个 Column Family 组成。Column Family 可以由任意多个 Column 组成，即 Column Family 支持动态扩展，无须预先定义 Column 的数量以及类型。所有 Column 均以二进制格式存储，用户需要自行进行类型转换。

Timestamp：每次数据操作对应的时间戳，可以看作是数据的版本号(Version Number)。

图 5-1 展示了一个 HBase 的数据模型实例，该表以反转的 URL 作为 Row Key。

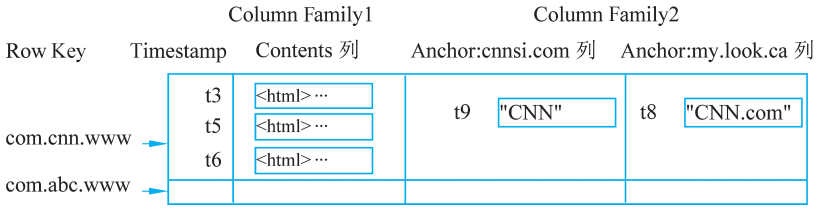

图 5-1 HBase 的数据模型

这个表格有两个 Column Family，分别是 Contents 和 Anchor。Contents 保存了页面的内容，每个时间戳对应一个页面，可以保留该页面的不同历史版本；Anchor 则保存了指向这个页面(即引用该页面)的其他页面的锚点(即其他页面的超链接指向本页面)的文本信息。

在这个实例中，CNN 的主页，被 Sports Illustrated(cnnsi.com)和 My Look(my.look.ca)两个页面指向，所以每行记录有 Anchor:cnnsi.com 和 Anchor:my.look.ca 两列，它们隶属于同一个 Column Family，即 Anchor。

这个表格有两个 Row Key，第一个 Row Key 保留了 3 个时间戳的 Contents 列，一个时间戳的 Anchor:cnnsi.com 列，以及另一个时间戳的 Anchor:my.look.ca 列。

当 Table 的记录数不断增加而变大后，一个 Table 逐渐分裂(Split)成多个 Split，称为 Region。每个 Region 由[startkey，endkey)或[startkey，endkey]表示，不同的 Region 被 Master 分配给相应的 RegionServer 进行管理(存储)，如图 5-2 所示。

HBase 中有两张特殊的 Table：-ROOT-和.META.。

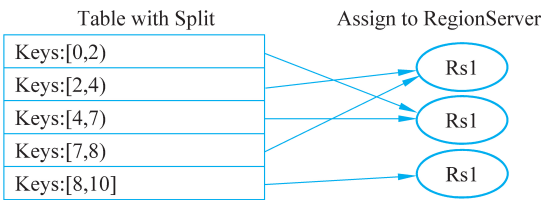

图 5-2　HBase 的 Region

（1）.META.：记录用户表的 Region 信息，其可以有多个 Region。

（2）-ROOT-：记录.META.表的 Region 信息，其只有一个 Region。

Zookeeper 中记录了-ROOT-表的位置信息（Location）。Client 访问数据之前需要先访问 Zookeeper，再访问-ROOT-表，然后访问.META.表，进而找到用户数据的位置，最后访问具体的数据。

5.4　HBase 系统架构

HBase 系统架构如图 5-3 所示，这是一个典型的主从（Master-Slave）架构，包含 HMaster、HRegionServer 和 Zookeeper。

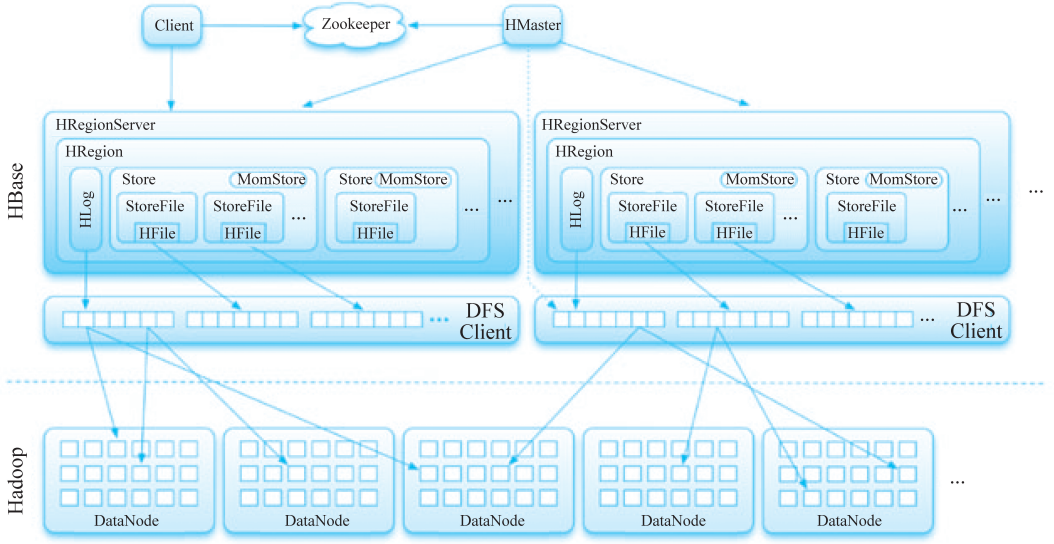

图 5-3　HBase 的系统架构

Client：HBase Client 使用 HBase 的 RPC 机制与 HMaster 和 HRegionServer 进行通信。对于管理类操作，Client 通过 RPC 访问 HMaster；对于数据读写类操作，Client 通过 RPC 访问 HRegionServer。

Zookeeper：Zookeeper Quorum 存储-ROOT-表的地址和 HMaster 的地址。HRegionServer 把自己以短暂的（Ephemeral）方式，注册到 Zookeeper 中，使 HMaster 可

以随时感知各个 HRegionServer 的健康状态。引入 Zookeeper，避免了 HMaster 的单点失败问题。

HMaster：HMaster 负责 Table 和 Region 的管理工作，具体包括如下内容。

（1）管理用户对 Table 的增加、删除、修改、查询操作。

（2）管理 HRegionServer 的负载均衡，调整 Region 分布。

（3）在 Region 分裂后，负责分配新的 Region。

（4）在 HRegionServer 停机后，负责失效 HRegionServer 上的 Region 的迁移。

HBase 可以启动多个 HMaster，通过 Zookeeper 的 Master Election 机制，保证总有一个 Active Master 运行，所以 HMaster 没有单点失败问题。

HRegionServer：主要负责响应用户 I/O 请求，向 HDFS 读写数据，是 HBase 中最核心的模块。每个 HRegionServer 大约可以管理 1000 个 Region。

图 5-4 为 HRegionServer 的内部结构。

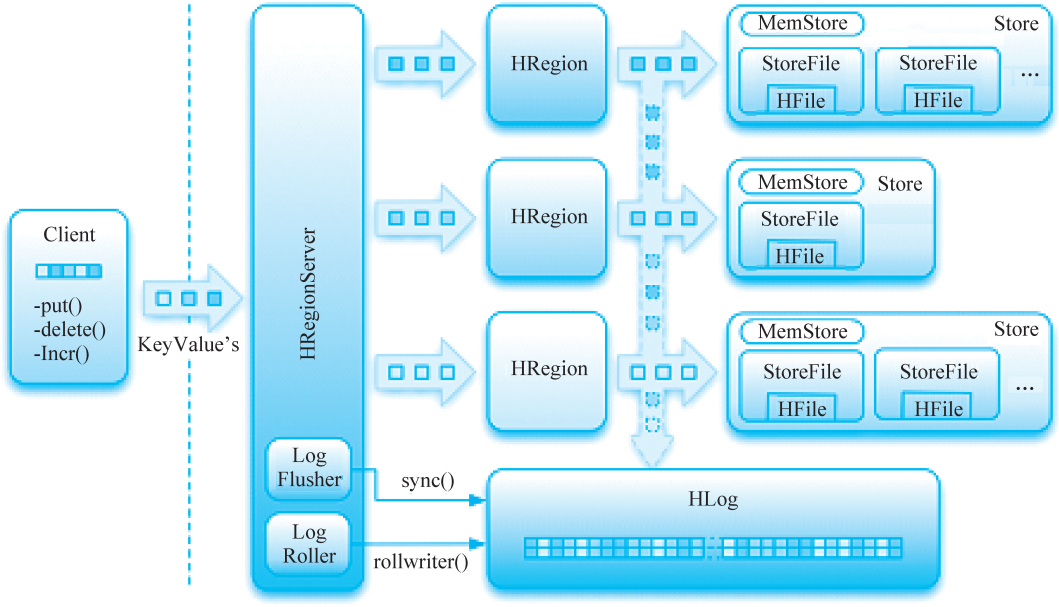

图 5-4　HRegionServer 的内部结构

HRegionServer 内部管理一系列 HRegion 对象，每个 HRegion 对应了 Table 中的一个 Region。HRegion 由多个 HStore 组成，每个 HStore 对应了 Table 中的一个 Column Family 的存储。可以看出，每个 Column Family 是一个集中的存储单元，因此最好将具备共同 I/O 特性的 Column 放在一个 Column Family 中，这样可以提高 I/O 效率。

HStore 存储是 HBase 存储的核心模块，由两部分组成，即 MemStore 和一系列 StoreFile。MemStore 是排序内存缓冲区（Sorted Memory Buffer），用户写入的数据首先会放入 MemStore，当 MemStore 满了以后，写入一个 StoreFile（底层实现是 HFile）。

当 StoreFile 数量增长到一定阈值，会触发压缩（Compact）合并操作，将多个 StoreFile 合并成一个大的 StoreFile，合并过程中会进行版本合并和数据删除。

HBase 只能增加数据，所有的更新和删除操作都是在后续的压缩(Compact)过程中进行的。这使得用户的写操作只要进入内存中就可以立即返回，保证了 HBase I/O 的高性能。

当 StoreFile 压缩后，会逐步形成越来越大的 StoreFile，当单个 StoreFile 大小超过一定阈值后，会触发分裂操作，把当前 Region 分裂成两个 Region，父 Region 会下线，新分裂出的两个子 Region 会被 HMaster 分配到相应的 HRegionServer 上，使得原先一个 Region 的读写压力得以分流到两个 Region 上，如图 5-5 所示。

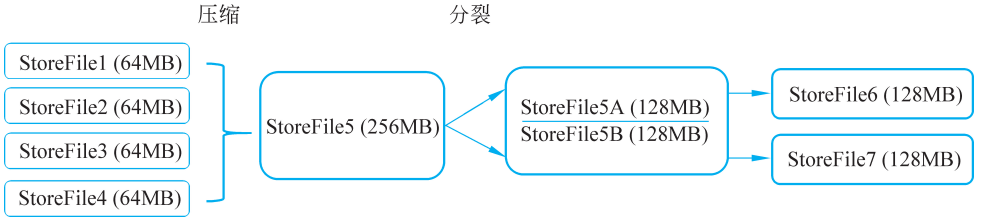

图 5-5　HBase 压缩/分裂过程

在分布式系统环境中，无法避免系统出错或者宕机。一旦 HRegionServer 意外退出，MemStore 中的内存数据将会丢失，因此需要引入 HLog。

每个 HRegionServer 都有一个 HLog 对象，HLog 是一个实现预写日志(Write Ahead Log，WAL)的类。在每次用户操作写入 MemStore 时，首先写一份数据到 HLog 文件中。HLog 文件定期删除旧的日志(已持久化到 StoreFile 中的数据)。

当 HRegionServer 意外终止后，HMaster 会通过 Zookeeper 感知到这个情况，HMaster 首先会处理遗留的 HLog 文件，将其中不同 Region 的 Log 数据进行拆分，分别放到相应 Region 的目录下，然后再将失效的 Region 重新分配。领取到这些 Region 的 HRegionServer，在装载 Region 的过程中，会发现有历史 HLog 需要处理，因此会重放 HLog(应用日志记录)中的数据到 MemStore 中，然后写入 StoreFile 中，完成数据恢复。

5.5　HBase 存储格式

HBase 的所有数据文件都存储在 Hadoop HDFS 上，包括两种文件类型。

(1) HFile：Hadoop 的二进制格式文件，实现 HBase 中 KeyValue 数据的存储。StoreFile 是对 HFile 做了轻量级包装，即 StoreFile 底层就是 HFile。

(2) HLog File：在 HBase 中实现 WAL(Write Ahead Log) 的存储格式，本质上是 Hadoop Sequence File。

HFile 的存储格式如图 5-6 所示。

HFile 文件是不定长的，长度固定的只有其中的两块：Trailer 和 FileInfo。Trailer 中有指针指向其他数据块的起始点。FileInfo 中记录了文件的一些元信息，如 AVG_KEY_LEN、AVG_VALUE_LEN、LAST_KEY、COMPARATOR、MAX_SEQ_ID_KEY 等。Data Index 和 Meta Index 块记录了每个 Data 块和 Meta 块的起始点。因此，HFile

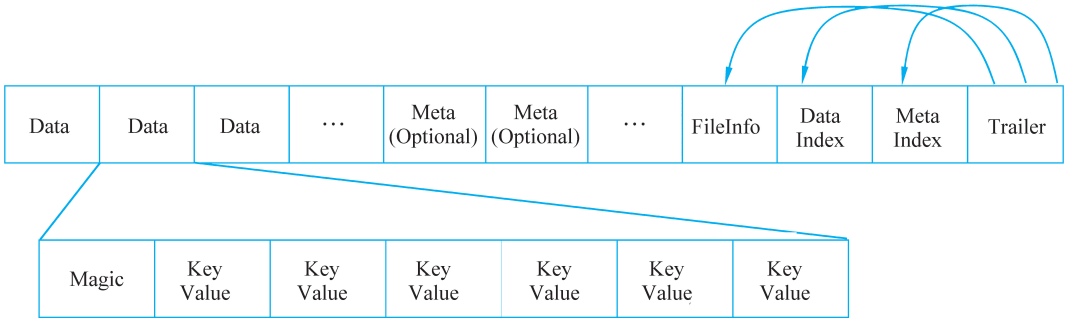

图 5-6　HFile 的存储格式

形成了一种自描述的文件结构。

Data 块是 HBase I/O 的基本单元。为了提高效率，HRegionServer 中实现了基于最近最少使用(Least Recently Used，LRU)的 Block Cache 机制。每个 Data 块的大小可以在创建一个 Table 时通过参数指定，较大的块有利于顺序扫描，较小的块有利于随机查询。

每个 Data 块除了开头的 Magic 信息以外，就是由一个个＜Key，Value＞拼接而成的。Magic 的内容是一些随机数字，目的是防止数据损坏。

HFile 里面的每个＜Key，Value＞都是一个简单的字节(Byte)数组，但是字节数组里面包含了很多项，并且有固定的结构。

如图 5-7 所示，字节数组开始是两个固定长度的数值，分别表示 Key 的长度和 Value 的长度。然后是 Key 部分具体项目排序如下：①固定长度的数值，表示 RowKey 的长度；②RowKey；③固定长度的数值，表示 Column Family 的长度；④Column Family(见图 5-1 中的 Anchor)；⑤Column Qualifier(见图 5-1 中的 cnnsi.com 或者 my.look.ca)；⑥两个固定长度的数值，表示 Timestamp 和 Key Type(Put/Delete)。最后 Value 部分结构简单，就是纯粹的二进制数据。

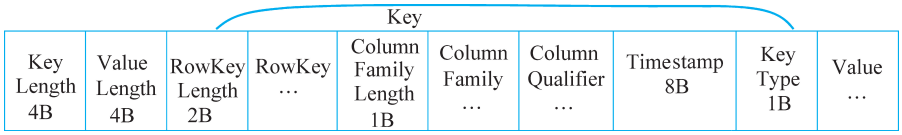

图 5-7　字节数组

HLogFile 的存储格式如图 5-8 所示。

HLog 文件就是一个普通的 Hadoop Sequence File。HLog Sequence File 的 Key 是 HLogKey 对象，HLogKey 中记录了写入数据的归属信息，除了 Table 和 Region 名字外，同时还包括 Sequence Number 和 Timestamp。Timestamp 是写入时间，Sequence Number 的起始值为 0，或者是最近一次存入文件系统的 Sequence Number。HLog Sequence File 的 Value 是 HBase 的＜Key，Value＞对象，即对应 HFile 中的 Key 和 Value。

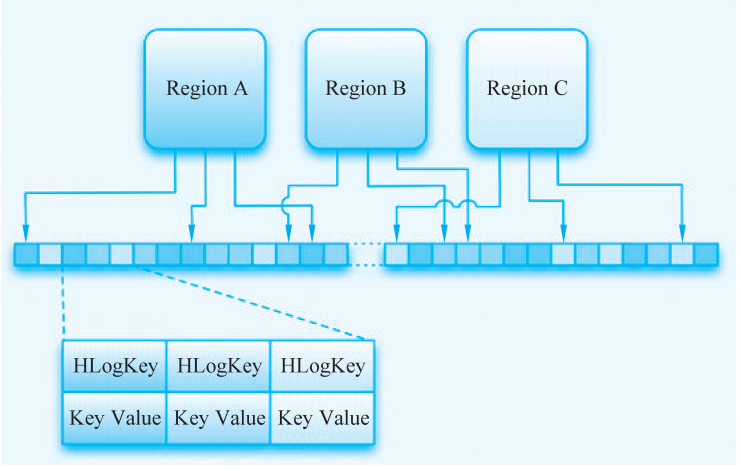

图 5-8　HLogFile 的存储格式

5.6　在 HBase 系统上运行 MapReduce

可以在 HBase 系统上运行 MapReduce 作业，如图 5-9 所示，实现数据的批处理。

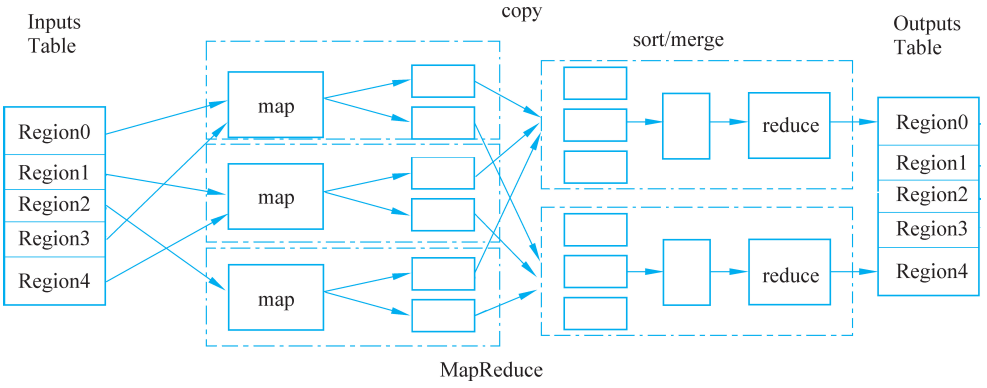

图 5-9　在 HBase 系统上运行 MapReduce 的原理

HBase Table 和 Region 的关系，类似于 HDFS File 和 Block 的关系。HBase 提供了配套的 TableInputFormat API 和 TableOutputFormat API，可以方便地将 HBase Table 作为 Hadoop MapReduce 的数据源（Source）和目的地（Sink）。对于 MapReduce 作业的应用开发人员，不需要关注 HBase 系统的细节。

5.7　HBase 安装、配置与运行

下面介绍如何安装、配置和运行 HBase 数据库。

配置 192.168.31.129、192.168.31.130、192.168.31.131 虚拟机的/etc/hosts 文件。增

加如下内容：

```
127.0.0.1 localhost
```

从官方网址下载 HBase 安装包 hbase-1.2.5-bin.tar.gz。

解压缩后，对目录进行重新命名。

```
cd /opt/linuxsir
tar xzvf hbase-1.2.5-bin.tar.gz
ls hbase-1.2.5

mv hbase-1.2.5 /opt/linuxsir/hbase
```

HBase 可以以独立（Standalone）、伪分布式（Pseudo-Distributed）、分布式（Distributed）3 种模式进行部署，这里只介绍分布式模式。

修改/opt/linuxsir/hbase/conf 目录下的 hbase-env.sh 脚本文件。设置 JAVA_HOME 的环境变量，以便启动 HBase。

```
export JAVA_HOME=/opt/linuxsir/java/jdk
export HBASE_CLASSPATH=/opt/linuxsir/hadoop/etc/hadoop

export HBASE_MANAGES_ZK=true
```

注意：

（1）HBASE_CLASSPATH 环境变量的设置，使得 HBase 和 Hadoop 的配置文件相连。也就是把 Hadoop 配置文件所在目录加入 HBASE_CLASSPATH 环境变量，使 HBase 看到 HDFS 的配置信息。

（2）如果想使用外部的 Zookeeper，那么 HBASE_MANAGES_ZK 应该设置为 false。

修改当前用户目录下的～/.bashrc 文件，并使之生效。

```
echo """>>~/.bashrc
echo "export HBASE_HOME=/opt/linuxsir/hbase">>~/.bashrc
echo "export PATH=\$HBASE_HOME/bin:\$PATH">>~/.bashrc
echo "export CLASSPATH=\$CLASSPATH:/opt/linuxsir/hbase/lib/*">>~/.bashrc

cat ~/.bashrc
source ~/.bashrc
```

执行如下命令，建立一些目录，包括 tmp/zookeeper 等。

```
mkdir /opt/linuxsir/hbase/tmp
```

```
ls /opt/linuxsir/hbase/zookeeper
//没有该目录就创建
mkdir /opt/linuxsir/hbase/zookeeper
```

有任何错误，可以先把 tmp 和 zookeeper 目录删除后再尝试。

```
rm -rf /opt/linuxsir/hbase/tmp
rm -rf /opt/linuxsir/hbase/zookeeper
```

编辑 HBase 配置文件，即/opt/linuxsir/hbase/conf 目录下的 hbase-site.xml，内容如下：

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
  <property>
    <name>hbase.rootdir</name><!-- 用户无需手动创建 HBase 的 data 目录，HBase
启动的时候会自动创建-->
    <value>hdfs://hd-master:9000/hbase</value>
  </property>

  <property>
    <name>hbase.zookeeper.property.dataDir</name>
    <value>/opt/linuxsir/hbase/zookeeper</value>
  </property>

  <property>
    <name>hbase.cluster.distributed</name>
    <value>true</value>
  </property>

  <property>
    <name>hbase.zookeeper.quorum</name>
    <value>hd-master,hd-slave1,hd-slave2</value>
  </property>

  <property>
    <name>hbase.zookeeper.property.dataDir</name>
    <value>/opt/linuxsir/hbase/zookeeper</value>
  </property>

  <property>
    <name>hbase.zookeeper.property.clientPort</name>
```

```
    <value>12181</value>
  </property>
</configuration>
```

在分布式模式下，HBase 集群包含若干节点，运行主 HMaster 进程、Backup HMaster 进程、Zookeeper 进程及 HRegionServer 进程等。

本书采用如下部署规划，注意 Zookeeper 进程的部署已经在 hbase-site.xml 文件中设置。

(1) HMaster 进程，运行在 hd-master、hd-slave1(backup)上。

(2) Zookeeper 进程，运行在 hd-master、hd-slave1、hd-slave2 上。

(3) HRegionServer 进程，运行在 hd-slave1、hd-slave2 上。

编辑/opt/linuxsir/hbase/conf 目录下的 regionservers 文件，内容如下：

```
hd-slave1
hd-slave2
```

编辑/opt/linuxsir/hbase/conf 目录下的 backup-masters 文件，内容如下：

```
hd-slave1
```

注意：这里没有关于 HMaster 的部署位置的配置。当在 hd-master 上启动 HBase 时，hd-master 节点将运行 HMaster 进程。

5.8 启动 HBase 并且测试

从 hd-master 复制/opt/linuxsir/hbase 目录以及～/.bashrc 文件到 hd-slave1 和 hd-slave2，并且在 3 个节点上使～/.bashrc 文件生效。

```
clear
scp -r /opt/linuxsir/hbase hd-slave1:/opt/linuxsir
scp -r /opt/linuxsir/hbase hd-slave2:/opt/linuxsir

scp ~/.bashrc hd-slave1:~/.bashrc
scp ~/.bashrc hd-slave2:~/.bashrc

source ~/.bashrc
ssh root@192.168.31.130 source ~/.bashrc
ssh root@192.168.31.131 source ~/.bashrc
```

启动 HBase 前，需要先启动 Hadoop。

首先，清空日志目录，以便启动出错时，方便查看最新出错信息。