实验5 部署全分布模式HBase集群和实战HBase

本实验的知识地图如图5-1所示（表示重点表示难点）。

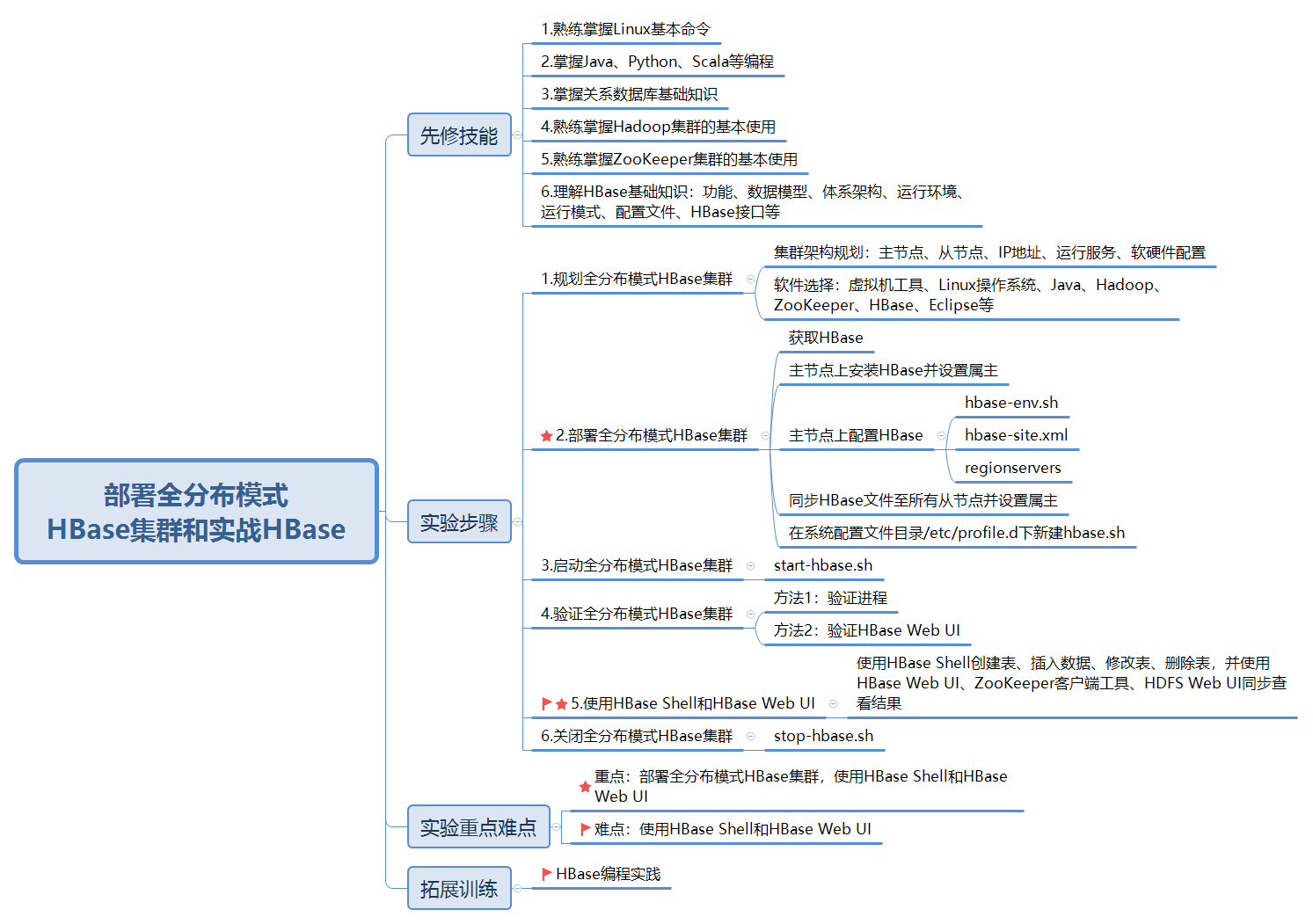


图5-1 实验5部署全分布模式HBase集群和实战HBase知识地图

一、实验目的

1. 理解HBase数据模型。

2. 理解HBase体系架构。

3. 熟练掌握HBase集群的部署。

4. 了解HBase Web UI的使用。

5. 熟练掌握HBase Shell常用命令的使用。

6. 了解HBase Java API，能编写简单的HBase程序。

二、实验环境

本实验所需的软件环境包括HDFS集群、ZooKeeper集群、HBase安装包、Eclipse。

三、实验内容

1. 规划全分布模式HBase集群。

2. 部署全分布模式HBase集群。

3. 启动全分布模式HBase集群。

4. 验证全分布模式HBase集群。

5. 使用HBase Web UI。

6. 使用HBase Shell常用命令。

7. 关闭全分布模式HBase集群。

四、实验原理

（一）初识HBase

传统的关系型数据库例如Oracle、MySQL擅长事务型数据，无法高效地存储和处理Web 3.0和大数据时代的多种非关系型数据，以Google的BigTable技术为代表的新型NoSQL数据库产品得到了飞速发展和应用，HBase数据库就是BigTable的开源实现，使用Java编写。作为Hadoop生态系统的重要组成部分之一，HBase是一个高可靠、高性能、列存储、可伸缩、实时读写的分布式数据库系统。

HBase利用Hadoop MapReduce来处理HBase中的海量数据，实现高性能计算；使用ZooKeeper作为协同服务，实现稳定服务和失败恢复；使用HDFS作为高可靠的底层存储，利用廉价集群提供海量数据存储能力。与Hadoop一样，HBase主要依靠横向扩展，通过不断增加廉价的商用服务器，来增加计算和存储能力。

HBase仅能通过行键（Row key）和行键的范围来检索数据，仅支持单行事务（可通过Hive支持来实现多表Join等复杂操作），主要用来存储非结构化和半结构化的松散数据。HBase的主要特点包括：数据稀疏、高维度（面向列）、分布式、键值有序存储、数据一致性。

（二）HBase数据模型

逻辑上，HBase以表的形式呈现给最终用户；物理上，HBase以文件的形式存储在HDFS中。为了高效管理数据，HBase设计了一些元数据库表来提高数据存取效率。

1. 逻辑模型

HBase以表（Table）的形式存储数据，每个表由行和列组成，每个列属于一个特定的列族（Column Family）。表中的行和列确定的存储单元称为一个元素（Cell），每个元素保存了同一份数据的多个版本，由时间戳（Time Stamp）来标识。行键（Row key）是数据行在表中的唯一标识，并作为检索记录的主键。在HBase中访问表中的行只有三种方式：通过单个行健访问、给定行键的范围扫描、全表扫描。行健可以是任意字符串，默认按字段顺序存储。表中的列定义为<family>:<qualifier>（<列族>:<限定符>），通过列族和限定符两部分可以唯一指定一个数据的存储列。元素由行健、列（<列族>:<限定符>）和时间戳唯一确定，元素中的数据以字节码的形式存储，没有类型之分。关于HBase逻辑模型中涉及到的相关概念如表5-1所示。

表5-1 HBase逻辑模型涉及相关概念说明

|  |  |
| --- | --- |
| 术语 | 说明 |
| 表（Table） | 由行和列组成，列划分为若干个列族 |
| 行键（Row key） | 每一行代表着一个数据对象，由行键来标识，行键会被建立索引，数据的获取通过Row key完成，采用字符串 |
| 列族（Column Family） | 列的集合，一个表中列可以分成不同列族，列族需在表创建时就定义好，数量不能太多，不能频繁修改 |
| 列限定符（Column Qualifier） | 表中具体一个列的名字，列族里的数据通过列限定符来定位，列限定符不用事先定义，也不需在不同行之间保持一致。被视为byte[]。列名以列族作为前缀，列族:列限定符 |
| 单元格（Cell） | 每一个行键、列族和列标识符共同确定的一个单元，存储在单元格里的数据称为单元格数据，单元格和单元格数据也没有特定的数据类型，以byte[]来存储 |
| 时间戳（Timestamp） | 每个单元格都保存着同一份数据的多个版本，这些版本采用时间戳进行索引。采用64位整型，降序存储 |

2. 物理模型

HBase是按照列存储的稀疏行/列矩阵，其物理模型实际上就是把逻辑模型中的一个行进行分割，并按照列族存储。

HBase中的所有数据文件都存储在Hadoop HDFS文件系统上，主要包括两种文件类型：HFile和HLog。

1）HFile

HFile是HBase中KeyValue数据的存储格式，是Hadoop的二进制格式文件，它是参考BigTable的SSTable和Hadoop的TFile的实现。从HBase开始到现在，HFile经历了三个版本，其中V2在0.92引入，V3在0.98引入。HFile V1版本在实际使用过程中发现它占用内存多，HFile V2版本针对此进行了优化，HFile V3版本基本和V2版本相同，只是在Cell层面添加了Tag数组的支持。鉴于此，编者主要针对V2版本进行分析，对V1和V3 版本感兴趣的读者可以查阅其它资料。

（1）HFile逻辑结构

HFile V2的逻辑结构如图5-2所示。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Scanned block section | Data Block | | | |
| … | | | |
| Leaf index block/Bloom block | | | |
| … | | | |
| Data Block | | | |
| … | | | |
| Leaf index block/Bloom block | | | |
| … | | | |
| Data Block | | | |
| Non-scanned block section | Meta block | … | Meta block | |
| Intermediate Level Data Index Blocks（optional） | | | |
| Load-on-open section | Root Data Index | | | Fields for midkey |
| Meta Index | | | |
| File Info | | | |
| Bloom filter metadata（interpreted by StoreFile） | | | |
| Trailer | Trailer fields | Version | | |

图5-2 HFile V2逻辑结构

从图5-2可以看出，HFile主要分为四个部分：数据块被扫描部分（Scanned block section），数据块不被扫描部分（Non-scanned block section），启动即加载部分（Load-on-open section）和HFile基本信息部分（Trailer）。

* Scanned block section：表示顺序扫描HFile时所有的数据块将会被读取，包括Leaf Index Block和Bloom Block。
* Non-scanned block section：表示在HFile顺序扫描的时候数据不会被读取，主要包括Meta Block和Intermediate Level Data Index Blocks两部分。
* Load-on-open section：这部分数据在HBase的region server启动时，需要加载到内存中，包括FileInfo、Bloom filter block、data block index和meta block index。
* Trailer：这部分主要记录了HFile的基本信息、各个部分的偏移值和寻址信息。

（2）HFile物理结构

HFile V2的物理结构如图5-3所示。

Root Data Index Block

HFileBlock

Data Block

…

Data Block

Leaf Index Block

Data Block

Meta Block

Meta Block

Intermediate Level Index Block

Trailer Block

Meta Index Block

File Info

Bloom Filter Metadata Block

Scanned block section

Non-scanned block section

Load-on-open section

Trailer

图5-3 HFile V2物理结构

如图5-3所示，HFlie会被切分为多个大小相等的block块，每个block的大小可以在创建表列族的时候通过参数blocksize进行指定，默认为64K，大号的Block有利于顺序Scan，小号Block有利于随机查询，因而需要权衡，而且所有block块都拥有相同的数据结构，如图左侧所示，HBase将block块抽象为一个统一的HFileBlock。HFileBlock支持两种类型，一种类型支持checksum，另一种不支持checksum。

2）HLog

HLog是HBase中WAL（Write Ahead Log）的存储格式，物理上是Hadoop的Sequence File。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sequenceid和write time，sequenceid的起始值为0，或者是最近一次存入文件系统中的sequenceid，write time是写入时间。HLog Sequece File的Value是HBase的KeyValue对象，即对应HFile中的KeyValue。HLog的逻辑结构如图5-4所示。

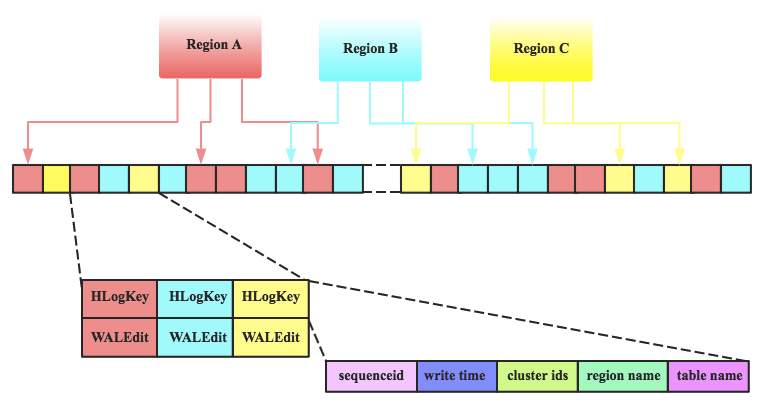


图5-4 HLog逻辑结构

3. 元数据表

从前文的HBase逻辑模型和物理模型的介绍可以看出，HBase的大部分操作都是在HRegionServer中完成，客户端想要进行插入、删除和查询数据都需要先找到对应的HRegionServer。客户端需要通过两个元数据表来找到HRegionServer和HRegion之间的对应关系：-ROOT-和.META.。它们是HBase的两张系统表，用于管理普通数据，其存储和操作方式和普通表相似，差别在于它们存储的是Region的分布情况和每个Region的详细信息，而不是普通数据。

HBase使用类似B+树的三层结构来保存Region位置信息，如图5-5所示，HBase的三层结构中各层次的名称和作用如表5-2所示。

ZooKeeper文件

-ROOT-表

.META.表

用户数据表

用户数据表

图5-5 HBase三层结构

表5-2 HBase的三层结构中各层次的名称和作用

|  |  |  |
| --- | --- | --- |
| 层次 | 名称 | 作用 |
| 第一层 | ZooKeeper文件 | 记录了-ROOT-表的位置信息 |
| 第二层 | -ROOT-表 | 记录了.META.表的Region位置信息，-ROOT-只能有一个Region，通过-ROOT-表就可以访问.META.表中的数据 |
| 第三层 | .META.表 | 记录了用户数据表的Region位置信息，.META.表可以有多个Region，保存了HBase中所有用户数据表的Region位置信息 |

（三）HBase体系架构

HBase采用Master/Slave架构，HBase集群成员包括Client、ZooKeeper集群、HMaster节点、HRegionServer节点，在底层HBase将数据存储于HDFS中。HBase的体系架构如图5-6所示。

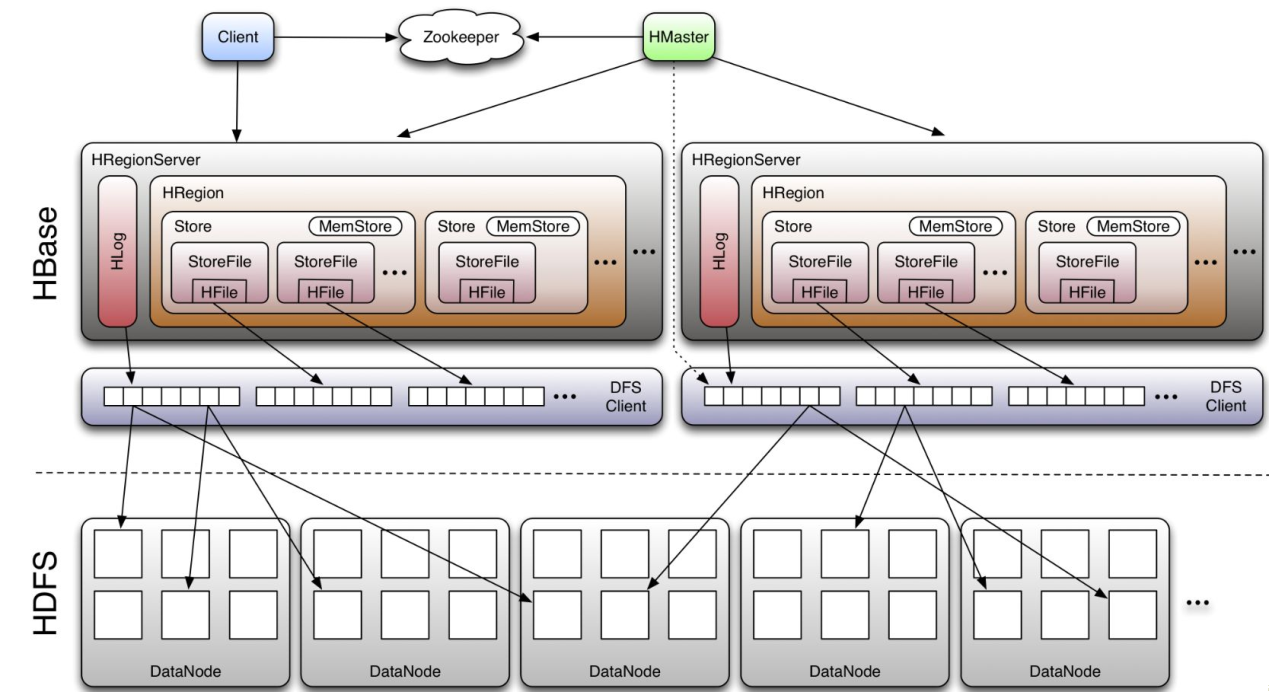


图5-6 HBase体系架构

1. 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与HMaster进行RPC；对于数据读写类操作，Client与HRegionServer进行RPC。客户端包含访问HBase的接口，通常维护一些缓存来加快HBase数据的访问速度，例如缓存各个Region的位置信息。

2. ZooKeeper

Zookeeper作为管理者，保证任何时候，集群中只有一个Master，对于HBase，ZooKeeper提供以下基本功能，如图5-7所示。

* 存储-ROOT-表的地址、HMaster的地址、HRegionServer的地址。
* 通过ZooKeeper，HMaster可以随时感知到各个HRegionServer的健康状态。
* ZooKeeper避免HMaster单点故障问题，HBase中可以启动多个HMaster，通过ZooKeeper的选举机制确保只有一个为当前HBase集群的master。

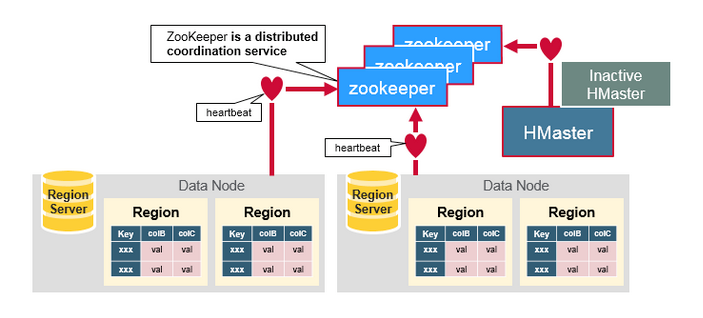


图5-7 ZooKeeper在HBase集群中功能

3. HMaster

HMaster是HBase的主服务程序，HBase中可以启动多个HMaster，通过Zookeeper选举机制保证每个时刻只有1个HMaster运行。HMaster主要完成以下任务，其功能如图5-8所示。

* 管理HRegionServer，实现其负载均衡。
* 管理和分配HRegion，比如在HRegion split时分配新的HRegion；在HRegionServer退出时迁移其内的HRegion到其他HRegionServer上。
* 实现DDL操作（Data Definition Language，namespace和table的增删改，column familiy的增删改等）。
* 管理namespace和table的元数据（实际存储在HDFS上）。
* 权限控制（ACL）。

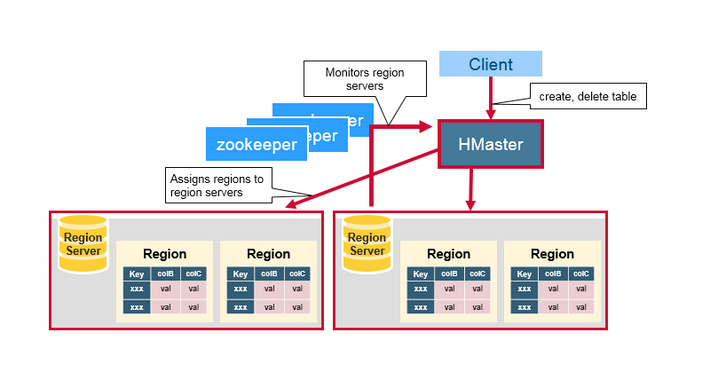


图5-8 HMaster功能

4. HRegionServer

HRegionServer是HBase的从服务程序，HBase集群中可以有多个HRegionServer，其主要功能包括以下几个方面。

* 存放和管理本地HRegion。
* 读写HDFS，管理Table中的数据。
* Client直接通过HRegionServer读写数据（从HMaster中获取元数据，找到RowKey所在的HRegion/HRegionServer后进行数据读写）。
* HRegionServer和DataNode一般会放在相同的Server上实现数据的本地化。

这里，还需要解释一下HRegion、Store、HLog的功能。

（1）HRegion

HRegionServer内部管理了一系列HRegion对象，每个HRegion对应了表中的一个Region，每个表最初只有一个Region，随着表中记录增加直到某个阈值，Region会被分割形成两个新的Region。HRegion的功能如图5-9所示。

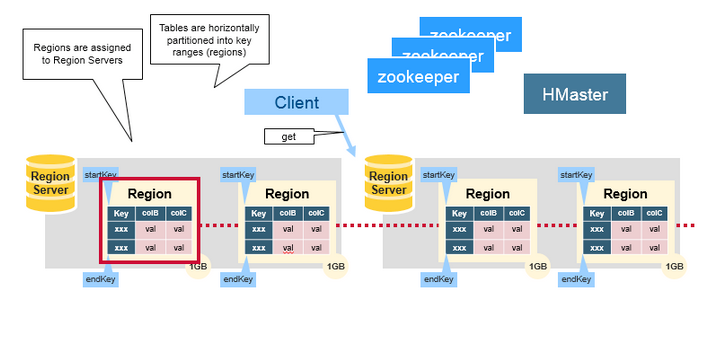


图5-9 HRegion功能

HRegion中由多个Store组成，每个Store对应了表中的一个Column Family的存储，可以看出每个Column Family其实就是一个集中的存储单元，因此最好将具有共同I/O特性的Column放在一个Column Family中，这样做最为高效。

（2）Store

Store是HBase存储的核心，由两部分组成：MemStore和StoreFiles。MemStore是有序内存缓冲区，用户写入的数据首先会放入MemStore，当MemStore满了以后会清空（Flush）成一个StoreFile（底层实现是HFile），当StoreFile文件数量增长到一定阈值，会触发合并（Compact）操作，将多个StoreFiles合并成一个StoreFile，合并过程中会进行版本合并和数据删除，因此可以看出HBase其实只有增加数据，所有的更新和删除操作都是在后续的Compact合并过程中进行的，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StoreFiles合并后，会逐步形成越来越大的StoreFile，当单个StoreFile大小超过一定阈值后，会触发分片（Split）操作，同时把当前Region分片成2个Region，父Region会下线，分片得到的2个孩子Region会被HMaster分配到相应的HRegionServer上，使得原先1个Region的压力得以分流到2个Region上。

（3）HLog

每个HRegionServer维护一个Hlog，而不是每个HRegion维护一个HLog。这样不同HRegion（来自不同表）的日志会混在一起，这样做的目的是不断追加单个文件相对于同时写多个文件而言，可以减少磁盘寻址次数，因此可以提高对表的写性能。但同时带来的麻烦是，如果一台HRegionServer下线，为了恢复其上的HRegion，需要将HRegionServer上的HLog进行拆分，然后分发到其它HRegionServer上进行恢复。

HLog文件定期会滚动出新的并删除旧的文件（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，HMaster首先会处理遗留的HLog文件，将其中不同HRegion的HLog数据进行拆分，分别放到相应HRegion的目录下，然后再将失效的HRegion重新分配，领取到这些HRegion的HRegionServer在加载Region的过程中会发现有历史HLog需要处理，因此会重做HLog中的数据到MemStore中，然后清空到StoreFiles，完成数据恢复。

（四）部署HBase

1. 运行环境

对于大部分Java开源产品而言，在部署与运行之前，总是需要搭建一个合适的环境，通常包括操作系统和Java环境两方面。同时，HBase依赖于ZooKeeper和HDFS，因此HBase部署与运行所需要的系统环境包括以下几个方面。

1）操作系统

HBase支持不同平台，在当前绝大多数主流的操作系统上都能够运行，例如Unix/Linux、Windows等。本书采用的操作系统为Linux发行版CentOS 7。

2）Java环境

HBase使用Java语言编写，因此它的运行环境需要Java环境的支持。

3）HDFS

HBase使用HDFS作为高可靠的底层存储，利用廉价集群提供海量数据存储能力，分布模式部署HBase时需要部署HDFS。

4）ZooKeeper

HBase使用ZooKeeper作为协同服务，实现稳定服务和失败恢复，因此需要部署ZooKeeper。

2. 运行模式

HBase运行模式有以下三种：

（1）单机模式（Standalone Mode）：只在一台计算机上运行，这种模式下，HBase所有守护进程包括Master、RegionServers和ZooKeeper都运行在一个JVM中，存储采用本地文件系统，没有采用分布式文件系统HDFS。

（2）伪分布模式（Pseudo-Distributed Mode）：只在一台计算机上运行，在这种模式下，HBase所有守护进程都运行在一个节点上，在一个节点上模拟了一个具有HBase完整功能的微型集群，存储采用分布式文件系统HDFS，但是HDFS的名称节点和数据节点都位于同一台计算机上。

（3）全分布模式（Fully-Distributed Mode）：在多台计算机上运行，在这种模式下，HBase的守护进程运行在多个节点上，形成一个真正意义上的集群，存储采用分布式文件系统HDFS，且HDFS的名称节点和数据节点位于不同计算机上。

3. 配置文件

HBase所有配置文件位于$HBASE\_HOME/conf下，如图5-10所示。

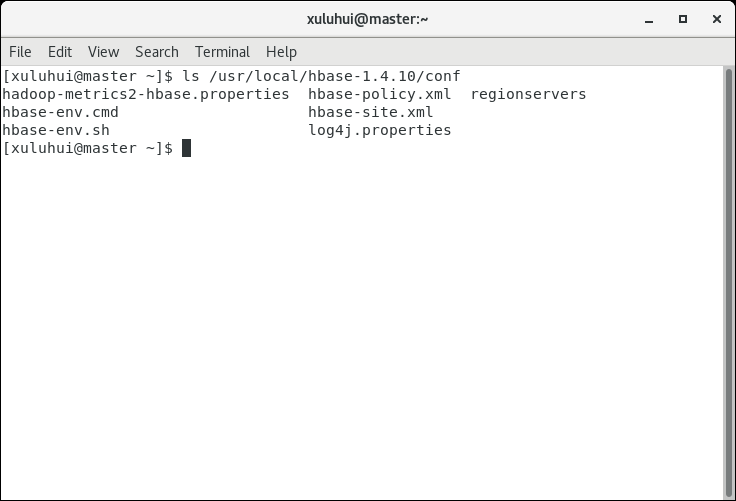


图5-10 HBase配置文件位置

HBase关键的几个配置文件的说明如表5-3所示，单机模式、伪分布模式和全分布模式下的HBase集群所需修改的配置文件有差异。关于HBase完整的配置文件介绍请参见官方文档https://hbase.apache.org/book.html#configuration。

表5-3 HBase配置文件（部分）

|  |  |
| --- | --- |
| 文件名称 | 描述 |
| hbase-env.sh | Bash脚本，设置Linux/Unix环境下运行HBase要用的环境变量，包括Java安装路径等 |
| hbase-site.xml | XML文件，HBase核心配置文件，包括HBase数据存放位置、ZooKeeper集群地址等配置项，其配置项会覆盖默认配置docs/hbase-default.xml |
| regionservers | 纯文本，设置运行HRegionServer从进程的机器列表，每行1个主机名 |

其中，配置文件hbase-site.xml中涉及的主要配置参数如表5-4所示。

表5-4 配置文件hbase-site.xml涉及的主要参数

|  |  |
| --- | --- |
| 参数 | 功能 |
| hbase.cluster.distributed | 指定HBase的运行模式，false是单机模式，true是分布式模式 |
| hbase.rootdir | 每个regionServer的共享目录，用来持久化HBase，默认为${hbase.tmp.dir}/hbase |
| hbase.zookeeper.quorum | Zookeeper集群的地址列表，用逗号分割，默认为localhost，是部署伪分布模式HBase集群用的 |
| hbase.zookeeper.property.dataDir | 与ZooKeeper的zoo.cfg中的配置参数dataDir一致 |

例如，单机模式的hbase-site.xml文件示例内容如下。

<configuration>

<property>

<name>hbase.rootdir</name>

<value>file:///home/testuser/hbase</value>

</property>

<property>

<name>hbase.zookeeper.property.dataDir</name>

<value>/home/testuser/zookeeper</value>

</property>

</configuration>

其中，配置项hbase.rootdir用于设置HBase和ZooKeeper数据的存放路径，前缀“file://” 表示本地文件系统。若HBase数据存放在HDFS上，需要设置hbase.rootdir指向HDFS，例如“hdfs://master:9000/hbase”。

（五）HBase接口

1. HBase Web UI

HBase集群主节点的Web UI地址为http://HMasterIP:16010，HBase集群从节点的Web UI地址为http://HRegionServerIP:16030。HBase主节点Web UI界面显示Master各种信息，包括Region Servers、Backup Masters、Tables、Tasks、Software Attributes，其效果如图5-11和图5-12所示。

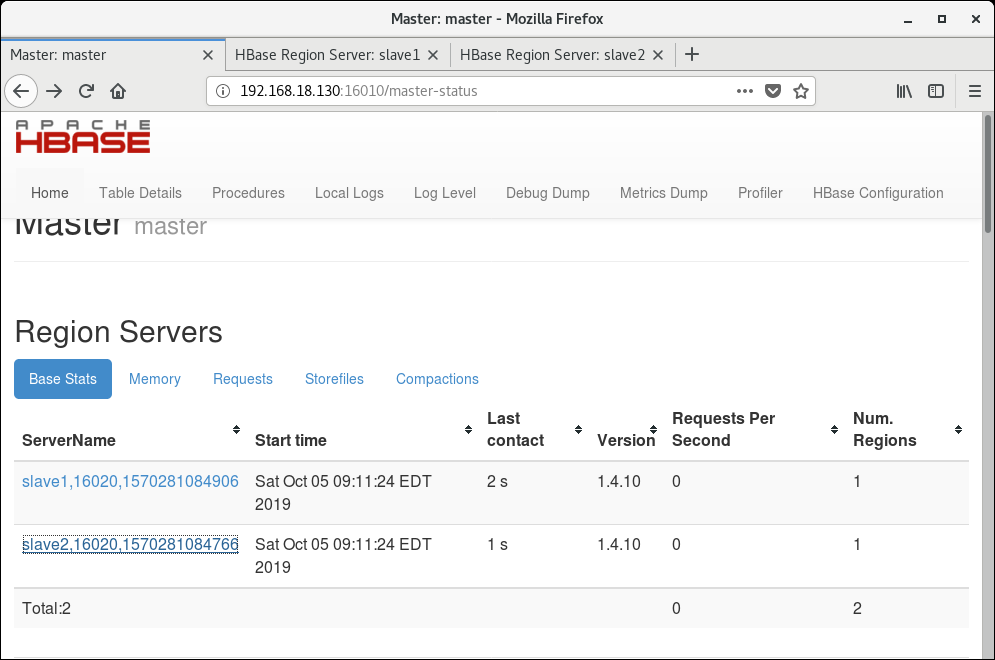


图5-11 HBase集群主节点Web UI运行效果图

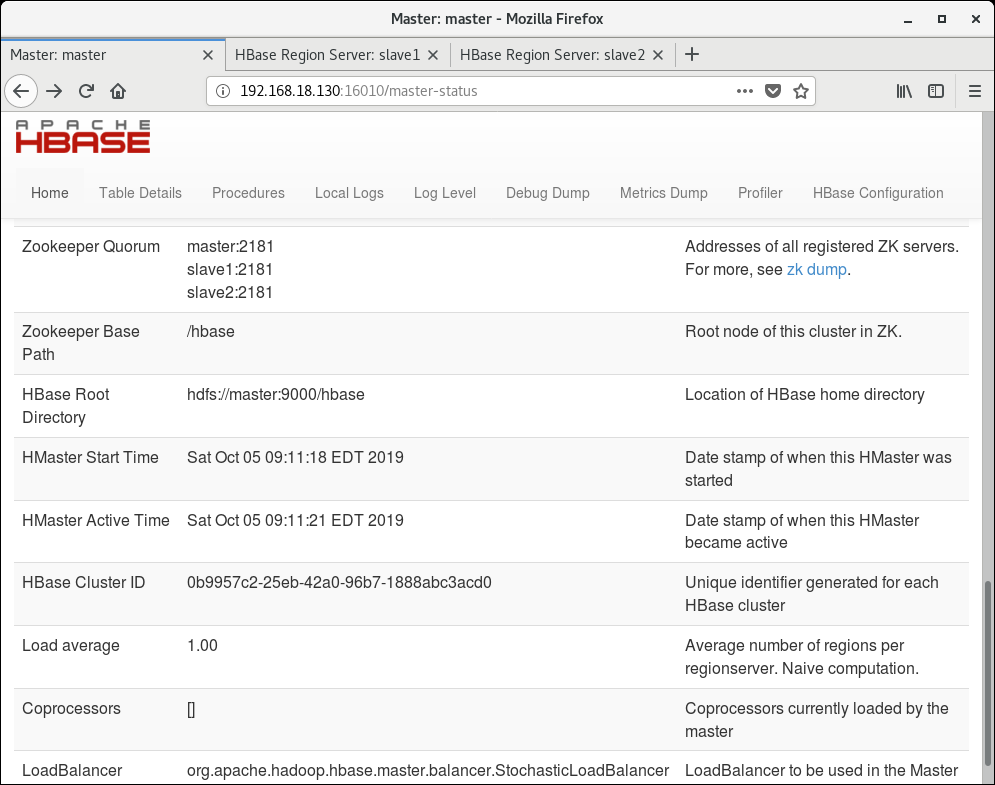


图5-12 HBase集群主节点Web UI中Software Attributes显示效果

HBase集群从节点Web UI界面显示RegionServer各种信息，包括Server Metrics、Block Cache、Tasks、Regions、Software Attributes，HBase集群从节点slave1的Web UI运行效果如图5-13所示。

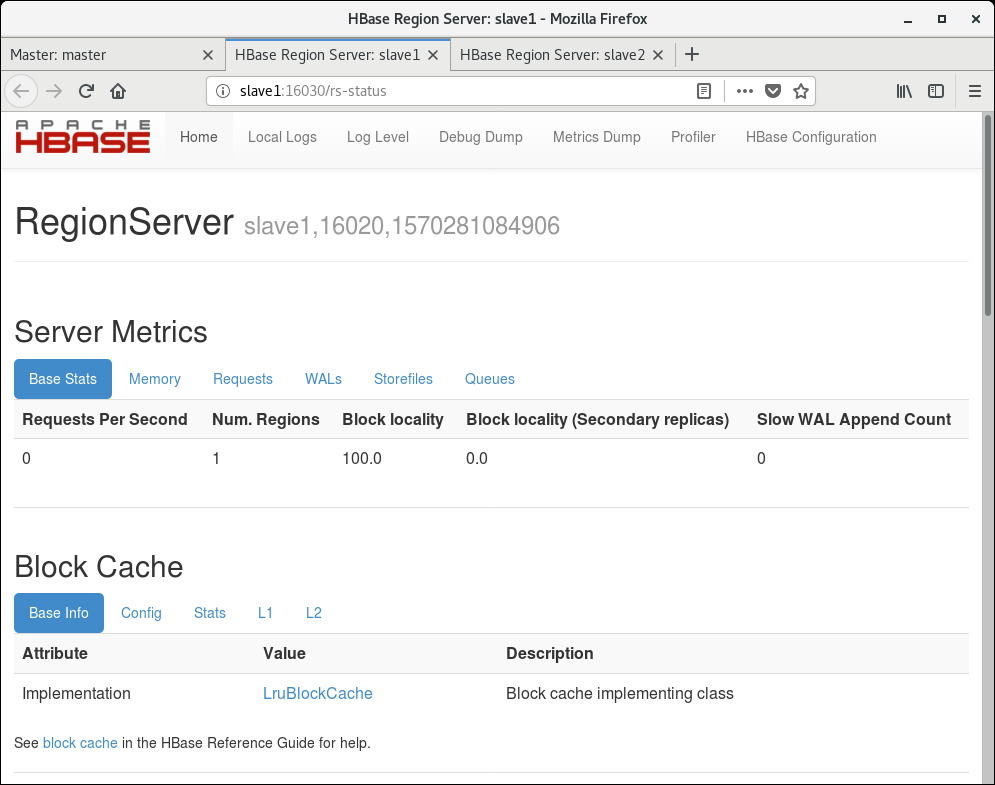


图5-13 HBase集群从节点slave1的Web UI运行效果图

2. HBase Shell

$HBASE\_HOME/bin下存放有HBase各种命令，如图5-14所示。

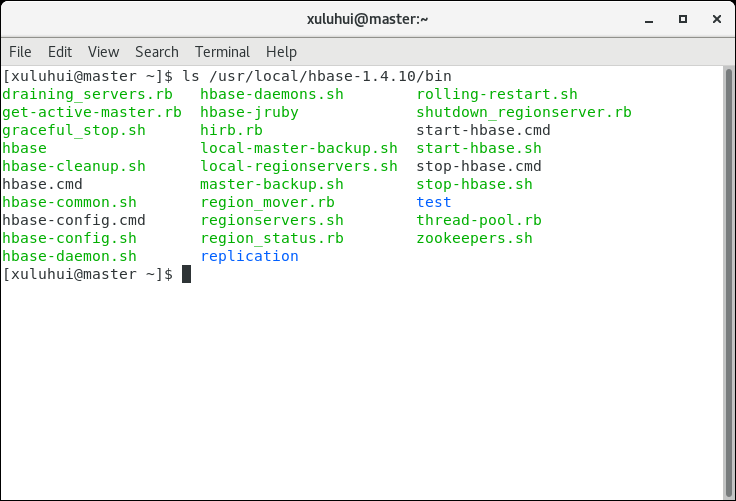


图5-14 HBase各种命令

其中，start-hbase.sh用于启动HBase集群，stop-hbase.sh用于关闭HBase集群。这里，我们详细介绍一下命令行工具“hbase shell”。

进入HBase命令行的入口命令是“bin/hbase shell”，进入后输入命令“help”可以查看HBase Shell命令的帮助信息，部分帮助信息如图5-15所示。

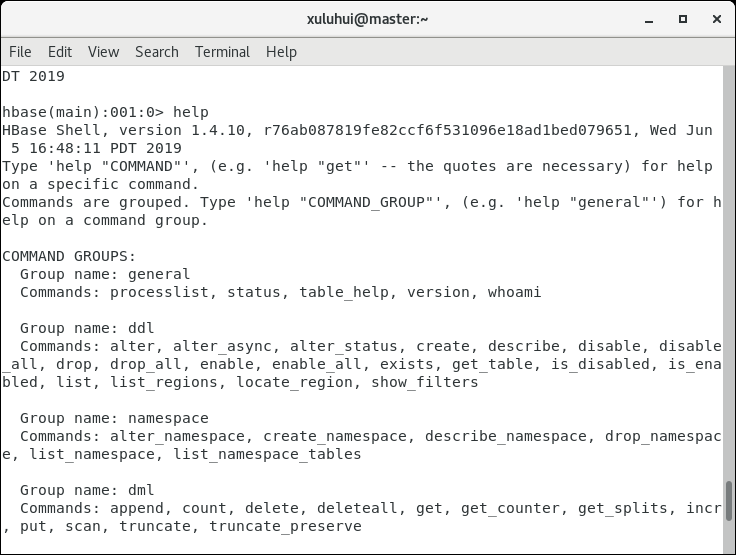


图5-15 HBase Shell帮助信息（部分）

从HBase Shell的帮助可以看出，HBase Shell命令共分为12大类，各类别所包含的命令如表5-5所示。

表5-5 HBase Shell命令

|  |  |
| --- | --- |
| 组名 | 包含命令 |
| general | processlist, status, table\_help, version, whoami |
| ddl | alter, alter\_async, alter\_status, create, describe, disable, disable\_all, drop, drop\_all, enable, enable\_all, exists, get\_table, is\_disabled, is\_enabled, list, list\_regions, locate\_region, show\_filters |
| namespace | alter\_namespace, create\_namespace, describe\_namespace, drop\_namespace, list\_namespace, list\_namespace\_tables |
| dml | append, count, delete, deleteall, get, get\_counter, get\_splits, incr, put, scan, truncate, truncate\_preserve |
| tools | assign, balance\_switch, balancer, balancer\_enabled, catalogjanitor\_enabled, catalogjanitor\_run, catalogjanitor\_switch, cleaner\_chore\_enabled, cleaner\_chore\_run, cleaner\_chore\_switch, clear\_deadservers, close\_region, compact, compact\_rs, compaction\_state, flush, is\_in\_maintenance\_mode, list\_deadservers, major\_compact, merge\_region, move, normalize, normalizer\_enabled, normalizer\_switch, split, splitormerge\_enabled, splitormerge\_switch, trace, unassign, wal\_roll, zk\_dump |
| replication | add\_peer, append\_peer\_tableCFs, disable\_peer, disable\_table\_replication, enable\_peer, enable\_table\_replication, get\_peer\_config, list\_peer\_configs, list\_peers, list\_replicated\_tables, remove\_peer, remove\_peer\_tableCFs, set\_peer\_bandwidth, set\_peer\_tableCFs, show\_peer\_tableCFs, update\_peer\_config |
| snapshots | clone\_snapshot, delete\_all\_snapshot, delete\_snapshot, delete\_table\_snapshots, list\_snapshots, list\_table\_snapshots, restore\_snapshot, snapshot |
| configuration | update\_all\_config, update\_config |
| security | grant, list\_security\_capabilities, revoke, user\_permission |
| procedures | abort\_procedure, list\_procedures |
| visibility labels | add\_labels, clear\_auths, get\_auths, list\_labels, set\_auths, set\_visibility |
| rsgroup | add\_rsgroup, balance\_rsgroup, get\_rsgroup, get\_server\_rsgroup, get\_table\_rsgroup, list\_rsgroups, move\_servers\_rsgroup, move\_servers\_tables\_rsgroup, move\_tables\_rsgroup, remove\_rsgroup, remove\_servers\_rsgroup |

3. HBase API

HBase API用于数据存储管理，主要操作包括建表、插入表数据、删除表数据、获取一行数据、表扫描、删除列族、删除表等。

HBase使用Java语言编写，提供了丰富了Java编程接口供开发人员调用，同时，HBase为其它多种编程语言也提供了API，C、C++、Scala、Python等。我们重点介绍HBase的Java客户端API使用方式，HBase部分核心Java API及其简单说明如表5-6所示。

表5-6 HBase部分核心Java API

|  |  |
| --- | --- |
| 类或接口 | 说明 |
| org.apache.hadoop.hbase.HBaseConfiguration | 用于管理HBase配置信息 |
| org.apache.hadoop.hbase.HBaseAdmin | 用于操作数据表的结构 |
| org.apache.hadoop.hbase.HTableDescriptor | 包含表的详细信息，例如表中的列族、该表的类型、该表是否只读、MemStore最大空间、Region什么时候应该分裂等 |
| org.apache.hadoop.hbase.HColumnDescriptor | 包含列族的详细信息，例如列族的版本号、压缩设置等 |
| org.apache.hadoop.hbase.HTable | 用于从HBase数据库中获取表信息或者更新 |
| org.apache.hadoop.hbase.client.Put | 用于对单元格执行添加数据操作 |
| org.apache.hadoop.hbase.client.Get | 用于获取单行的信息 |
| org.apache.hadoop.hbase.client.Scan | 限定需要查找的数据，例如限定版本号、起始行号、终止行号、列族、列限定符、返回值的数量上限等 |
| org.apache.hadoop.hbase.client.Result | 用于存放Get和Scan操作后的查询结果，并以<key，value>格式存储在map结构中 |
| org.apache.hadoop.hbase.client.ResultScanner | 客户端获取值的接口 |

关于HBase API的详细资料读者请参考本地文件file:///usr/local/hbase-1.4.10/docs/apidocs/index.html或者官网https://hbase.apache.org/apidocs/index.html。Apache HBase 1.4.10 API的首页如图5-16所示。

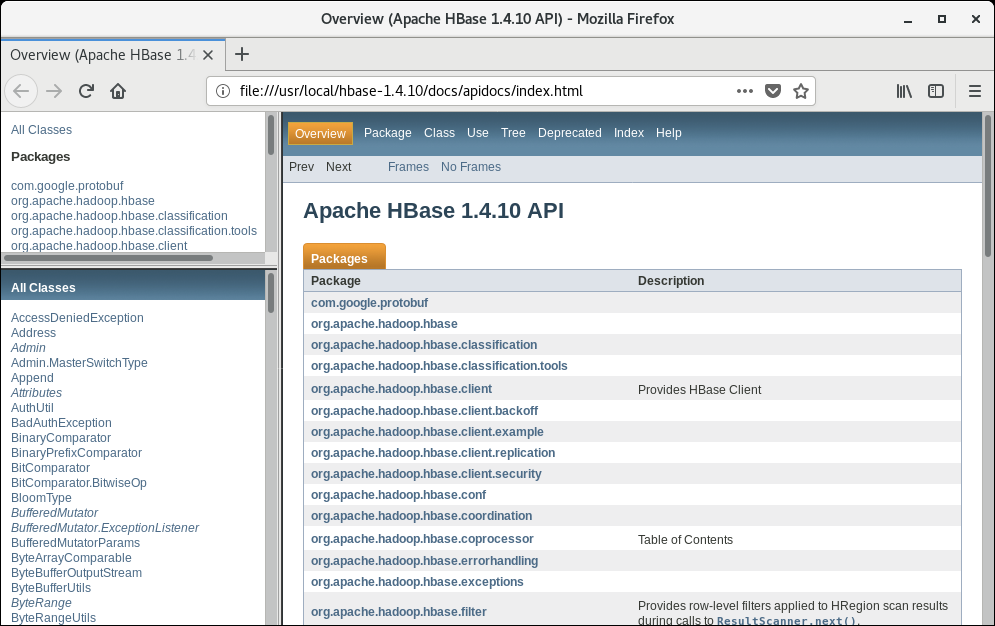


图5-16 Apache HBase 1.4.10 API官方参考指南首页

五、实验步骤

（一）规划全分布模式HBase集群

1. HBase集群规划

本实验拟采用全分布模式部署HBase，将HBase集群运行在Linux上，使用3台安装有Linux操作系统的机器，主机名分别为master、slave1、slave2。具体全分布模式HBase集群规划表如表5-7所示。

表5-7 全分布模式HBase集群部署规划表

|  |  |  |  |
| --- | --- | --- | --- |
| 主机名 | IP地址 | 运行服务 | 软硬件配置 |
| master | 192.168.18.130 | NameNode  SecondaryNameNode  QuorumPeerMain  HMaster | 内存：4G  CPU：1个2核  硬盘：40G  操作系统：CentOS 7.6.1810  Java：Oracle JDK 8u191  Hadoop：Hadoop 2.9.2  ZooKeeper：ZooKeeper 3.4.13  HBase：HBase 1.4.10  Eclipse：Eclipse IDE 2018-09 for Java Developers |
| slave1 | 192.168.18.131 | DataNode  QuorumPeerMain  HRegionServer | 内存：1G  CPU：1个1核  硬盘：20G  操作系统：CentOS 7.6.1810  Java：Oracle JDK 8u191  Hadoop：Hadoop 2.9.2  ZooKeeper：ZooKeeper 3.4.13  HBase：HBase 1.4.10 |
| slave2 | 192.168.18.132 | DataNode  QuorumPeerMain  HRegionServer | 内存：1G  CPU：1个1核  硬盘：20G  操作系统：CentOS 7.6.1810  Java：Oracle JDK 8u191  Hadoop：Hadoop 2.9.2  ZooKeeper：ZooKeeper 3.4.13  HBase：HBase 1.4.10 |

2. 软件选择

本实验中所使用各种软件的名称、版本、发布日期及下载地址如表5-8所示。

表5-8 本实验使用的软件名称、版本、发布日期及下载地址

|  |  |  |  |
| --- | --- | --- | --- |
| 软件名称 | 软件版本 | 发布日期 | 下载地址 |
| VMware Workstation Pro | VMware Workstation 14.5.7 Pro for Windows | 2017年6月22日 | https://www.vmware.com/products/workstation-pro.html |
| CentOS | CentOS 7.6.1810 | 2018年11月26日 | https://www.centos.org/download/ |
| Java | Oracle JDK 8u191 | 2018年10月16日 | http://www.oracle.com/technetwork/java/javase/downloads/index.html |
| Hadoop | Hadoop 2.9.2 | 2018年11月19日 | http://hadoop.apache.org/releases.html |
| ZooKeeper | ZooKeeper 3.4.13 | 2018年7月15日 | http://zookeeper.apache.org/releases.html |
| HBase | HBase 1.4.10 | 2019年6月10日 | https://hbase.apache.org/downloads.html |
| Eclipse | Eclipse IDE 2018-09 for Java Developers | 2018年9月 | https://www.eclipse.org/downloads/packages/ |

注意，本书采用的是HBase版本是1.4.10，3个节点的机器名分别为master、slave1、slave2，IP地址依次为192.168.18.130、192.168.18.131、192.168.18.132，后续内容均在表5-8规划基础上完成，请读者务必确认自己的HBase版本、机器名等信息。

（二）部署全分布模式HBase集群

HBase目前有1.x和2.x两个系列的版本，建议读者使用当前的稳定版本，本书采用稳定版本HBase 1.4.10，因此本章的讲解都是针对这个版本进行的。尽管如此，由于HBase各个版本在部署和运行方式上的变化不大，因此本章的大部分内容都适用于HBase其他版本。

1. 初始软硬件环境准备

（1）准备3台机器，安装操作系统，编者使用CentOS Linux 7。

（2）对集群内每一台机器，配置静态IP、修改机器名、添加集群级别域名映射、关闭防火墙。

（3）对集群内每一台机器，安装和配置Java，要求Java 1.7或更高版本，编者使用Oracle JDK 8u191。

（4）安装和配置Linux集群中主节点到从节点的SSH免密登录。

（5）在Linux集群上部署全分布模式Hadoop集群。

（6）在Linux集群上部署ZooKeeper集群。

以上步骤编者已在教材实验1、实验4中详细介绍，具体操作过程请读者参见教材，此处不再赘述。

2. 获取HBase

HBase官方下载地址为https://hbase.apache.org/downloads.html，建议读者下载stable目录下的当前稳定版本。编者采用的HBase稳定版本是2019年6月10日发布的HBase 1.4.10，其安装包文件hbase-1.4.10-bin.tar.gz例如存放在master机器的/home/xuluhui/Downloads中。

3. 主节点上安装HBase并设置属主

（1）在master机器上，切换到root，解压hbase-1.4.10-bin.tar.gz到安装目录如/usr/local下，依次使用的命令如下所示。

su root

cd /usr/local

tar -zxvf /home/xuluhui/Downloads/hbase-1.4.10-bin.tar.gz

（2）为了在普通用户下使用HBase，将HBase安装目录的属主设置为Linux普通用户例如xuluhui，使用以下命令完成。

chown -R xuluhui /usr/local/hbase-1.4.10

4. 主节点上配置HBase

HBase所有配置文件位于$HBASE\_HOME/conf下，具体的配置文件如前文图5-9所示。本实验中编者仅修改hbase-env.sh、hbase-site.xml、regionservers三个配置文件。

假设当前目录为“/usr/local/hbase-1.4.10”，切换到普通用户如xuluhui下，在主节点master上配置HBase的具体过程如下所示。

1）编辑配置文件hbase-env.sh

hbase-env.sh用于设置Linux/Unix环境下运行HBase要用的环境变量，包括Java安装路径等，使用“vim conf/hbase-env.sh”对其进行如下修改。

（1）设置JAVA\_HOME，与master上之前安装的JDK位置、版本一致，将第27行的注释去掉，并修改为以下内容，修改后的效果如图5-17所示。

export JAVA\_HOME=/usr/java/jdk1.8.0\_191/

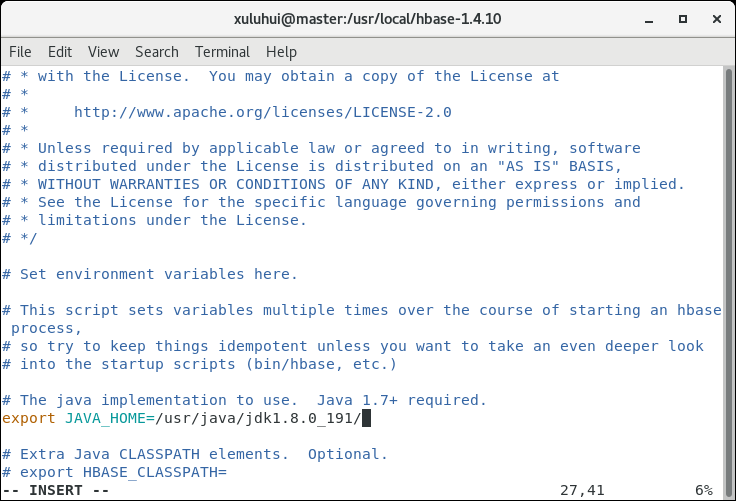


图5-17 编辑配置文件hbase-env.sh中JAVA\_HOME

（2）将第46、47行的PermSize作为注释，因为JDK8中无需配置，修改后的效果如图5-18所示。

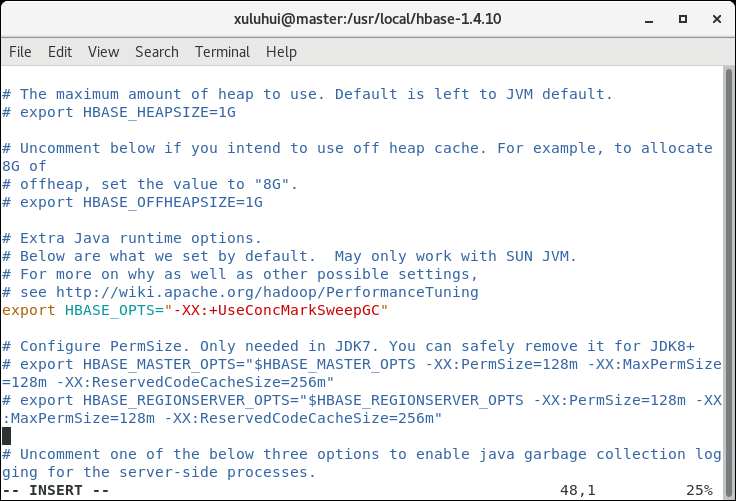


图5-18 编辑配置文件hbase-env.sh中PermSize

JDK8下若PermSize配置不作为注释或删掉，则启动HBase集群时会出现以下“warning”警告信息，如图5-19所示。

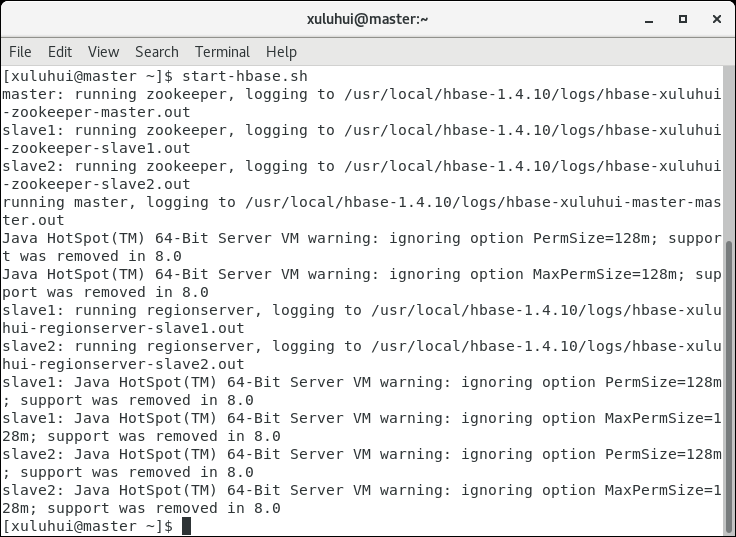


图5-19 JDK8下不注释掉PermSize配置启动HBase集群出现的警告信息

（3）设置HBASE\_PID\_DIR，修改进程号文件的保存位置，该参数默认为“/tmp”，将第120行修改为以下内容，如图5-20所示。其中pids目录由HBase集群启动后自动创建。

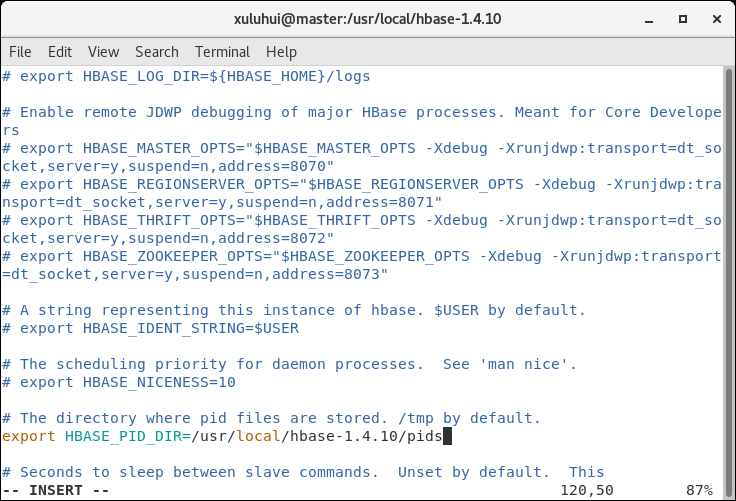


图5-20 编辑配置文件hbase-env.sh中HBASE\_PID\_DIR

（4）设置HBASE\_MANAGES\_ZK，将其值设置为false，即关闭HBase本身的ZooKeeper集群，将第128行修改为以下内容，如图5-21所示。

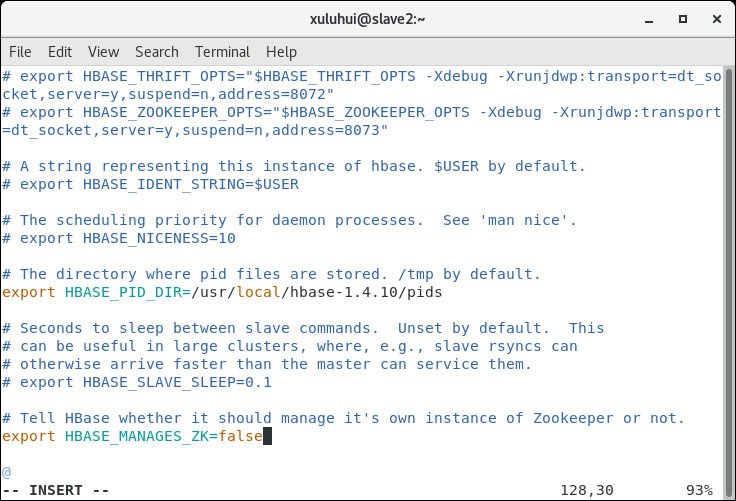


图5-21 编辑配置文件hbase-env.sh中HBASE\_MANAGES\_ZK

2）编辑配置文件hbase-site.xml

hbase-site.xml是HBase核心配置文件，包括HBase数据存放位置、ZooKeeper集群地址等配置项。在master机器上修改配置文件hbase-site.xml，具体内容如下所示。

<configuration>

<!-- 每个regionServer的共享目录，用来持久化HBase，默认为${hbase.tmp.dir}/hbase -->

<property>

<name>hbase.rootdir</name>

<value>hdfs://master:9000/hbase</value>

</property>

<!-- HBase集群模式，false表示HBase的单机模式，true表示是分布式模式，默认为false -->

<property>

<name>hbase.cluster.distributed</name>

<value>true</value>

</property>

<!-- HBase依赖的ZooKeeper集群地址，默认为localhost -->

<property>

<name>hbase.zookeeper.quorum</name>

<value>master:2181,slave1:2181,slave2:2181</value>

</property>

</configuration>

3）编辑配置文件regionservers

Regionservers用于设置运行HRegionServer从进程的机器列表，每行1个主机名。在master机器上修改配置文件regionservers，该文件原来内容为“localhost”，修改为以下内容。

slave1

slave2

5. 同步HBase文件至所有从节点并设置属主

（1）使用scp命令将master机器中目录“hbase-1.4.10”及下属子目录和文件统一拷贝至集群中所有HBase从节点上，例如slave1和slave2上，依次使用的命令如下所示。

scp -r /usr/local/hbase-1.4.10 root@slave1:/usr/local/hbase-1.4.10

scp -r /usr/local/hbase-1.4.10 root@slave2:/usr/local/hbase-1.4.10

（2）依次将所有HBase从节点slave1、slave2上的HBase安装目录的属主也设置为Linux普通用户例如xuluhui，使用以下命令完成。

chown -R xuluhui /usr/local/hbase-1.4.10

读者可以使用ssh命令直接在master节点上远程连接所有HBase从节点，以修改HBase安装目录的属主。以slave2为例，依次使用的命令及效果如图5-22所示。

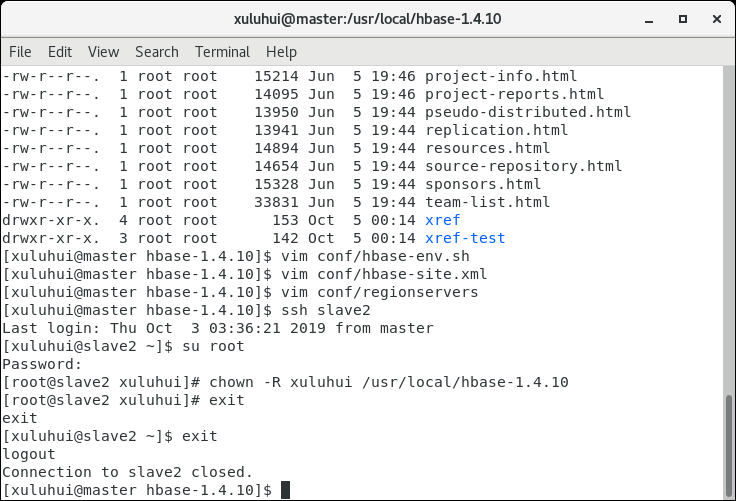


图5-22 通过ssh直接在master节点上修改slave2上HBase安装目录属主

至此，Linux集群中各个节点的HBase均已安装和配置完毕。

6. 在系统配置文件目录/etc/profile.d下新建hbase.sh

另外，为了方便使用HBase各种命令，可以在HBase集群所有机器上使用“vim /etc/profile.d/hbase.sh”命令在/etc/profile.d文件夹下新建文件hbase.sh，并添加如下内容。

export HBASE\_HOME=/usr/local/hbase-1.4.10

export PATH=$HBASE\_HOME/bin:$PATH

重启机器，使之生效。

此步骤可省略，之所以将$HBASE\_HOME/bin目录加入到系统环境变量PATH中，是因为当输入启动和管理HBase集群命令时，无需再切换到$HBASE\_HOME/bin目录，否则会出现错误信息“bash: \*\*\*\*: command not found...”。

（三）启动全分布模式HBase集群

1. 启动HDFS集群

在主节点上使用命令“start-dfs.sh”启动HDFS集群，使用的命令及运行效果如图5-23所示，从图5-23中可以看出，HDFS主进程NameNode成功启动，slave1和slave2上的从进程DataNode此处未展示，读者应保证HDFS所有主从进程都启动成功。

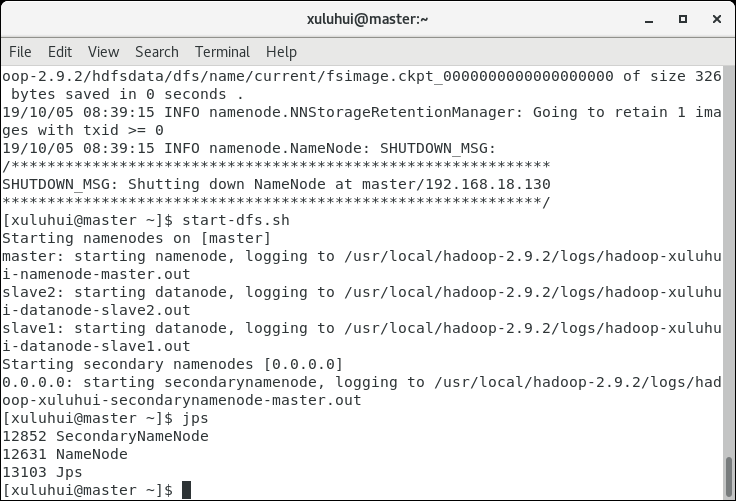


图5-23 启动HDFS集群

2. 启动ZooKeeper集群

由于本实验中HBase并未自动管理ZooKeeper，所以用户需要手工启动ZooKeeper集群。在ZooKeeper集群的所有节点上使用命令“zkServer.sh start”启动ZooKeeper集群，编者为了方便，在节点master上使用ssh远程连接slave1、slave2，完成了各个节点ZooKeeper的启动工作，依次使用的命令和执行效果如图5-24所示。

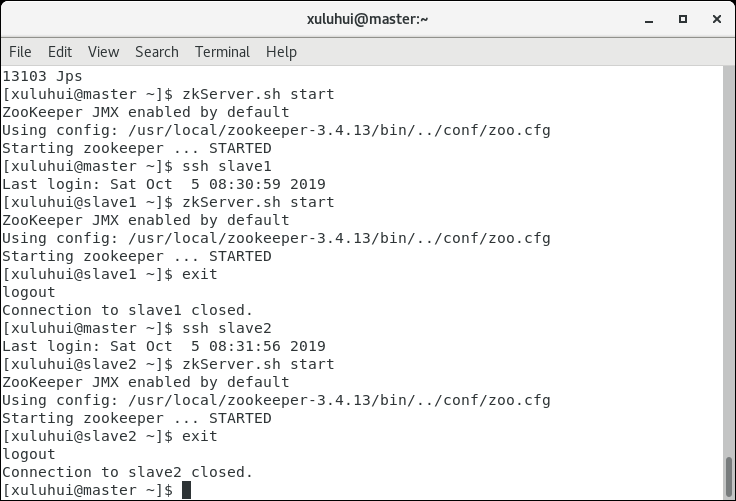


图5-24 启动ZooKeeper集群

此处未展示ZooKeeper各进程和各节点状态，读者应保证ZooKeeper集群成功启动，可以使用命令jps命令验证进程，使用命令“zkServer.sh status”查看状态。

3. 启动HBase集群

在主节点上启动HBase集群，使用的命令及执行效果如图5-25所示。

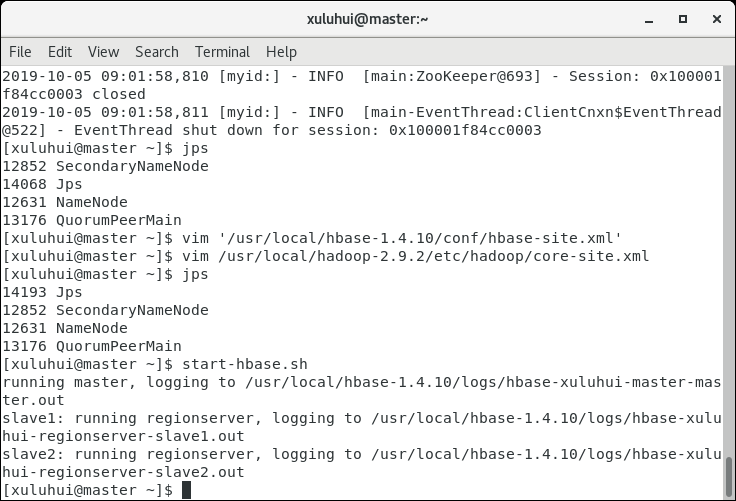


图5-25 启动HBase集群

（四）验证全分布模式HBase集群

启动HBase集群后，可通过以下两种方法验证HBase集群是否成功部署。

**1. 验证进程（方法1）**

使用命令jps查看，按本实验设置，HBase主节点master上应该有HBase主进程HMaster、HDFS主进程NameNode、ZooKeeper进程QuorumPeerMain，HBase从节点slave1、slave2上应该有HBase从进程HRegionServer、HDFS从进程DataNode、ZooKeeper进程QuorumPeerMain，效果如图5-26所示。



图5-26 使用jps验证HBase集群是否部署成功

启动HBase主进程HMaster和从进程HRegionServer的同时，会依次在集群的主从节点$HBASE\_HOME下自动生成pids目录及其下HBase进程号文件\*.pid和ZooKeeper节点文件\*.znode，效果如图5-27所示的文件。

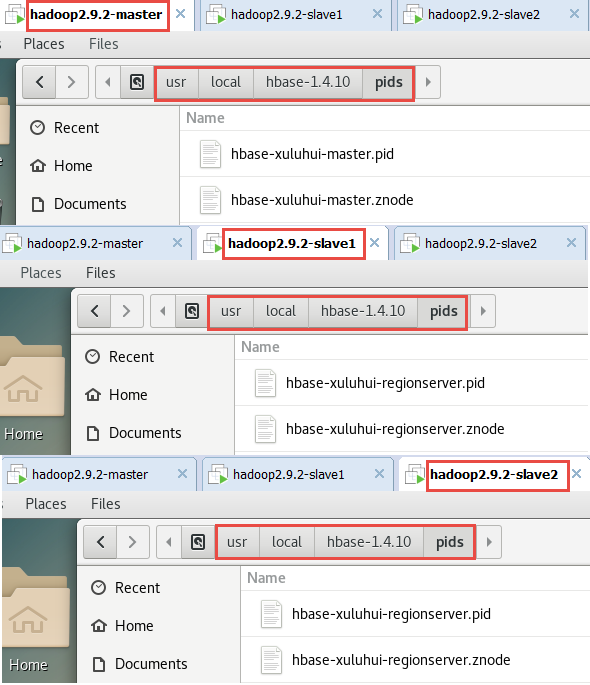


图5-27 当前HBase集群主从节点上目录“/usr/local/hbase-1.4.10/pids”中文件

另外，启动HBase主进程HMaster和从进程HRegionServer的同时，会依次在集群的主从节点$HBASE\_HOME下自动生成logs目录及其下日志文件\*.log等，效果如图5-28所示的文件。

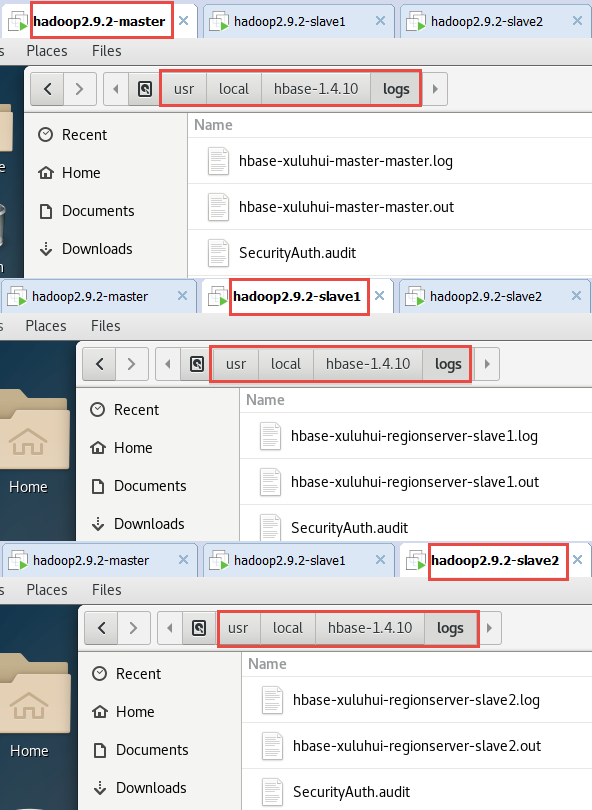


图5-28 当前HBase集群主从节点上目录“/usr/local/hbase-1.4.10/logs”中文件

2. 验证HBase Web UI（方法2）

打开浏览器，输入HBase集群主节点Web UI地址http://192.168.18.130:16010，效果如上文图5-11和图5-12所示。同时打开HBase集群从节点Web UI地址http://192.168.18.132:16030、http://192.168.18.133:16030，其中HBase集群从节点slave1的Web UI运行效果图如图5-13所示。若主、从节点的Web UI都能够顺利打开，则表示全分布式的HBase集群部署成功。

（五）使用HBase Shell和HBase Web UI

【案例5-1】使用HBase Shell命令在HBase下建立一个student表，其逻辑模型如表5-9所示，对该表进行添加数据、修改列族模式等操作后，使用HBase Web UI、ZooKeeper、HDFS Web UI等工具查看该表，最后删除该表。

表5-9 HBase表student逻辑模型

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Row key | 列族personalInfo | | | | 列族gradeInfo | | |
| SNO | Sname | Ssex | Sage | Sdept | BigData | Math | English |
|  |  |  |  |  |  |  |  |

（1）使用命令“hbase shell”进入HBase命令行。使用命令及执行结果如图5-29所示。

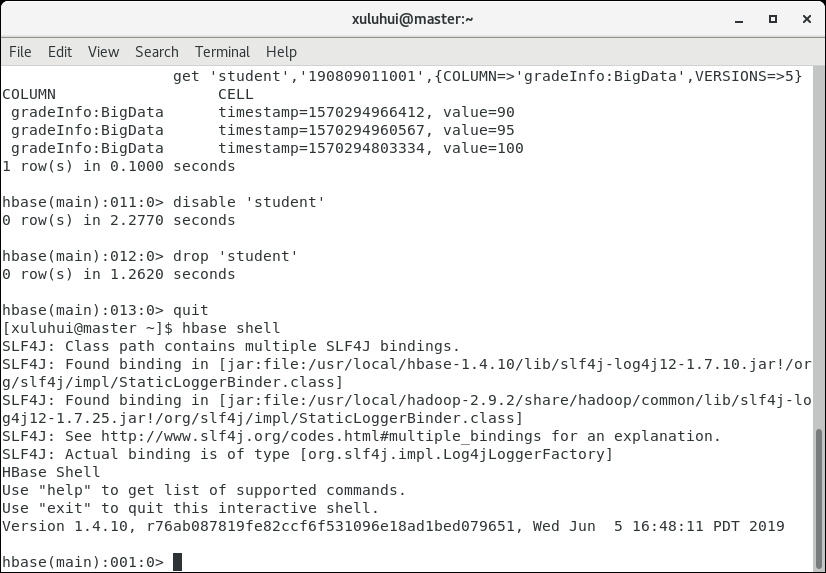


图5-29 进入HBase Shell

（2）使用create命令创建HBase表student，然后使用describe查看表结构。使用命令及执行结果如图5-30所示。

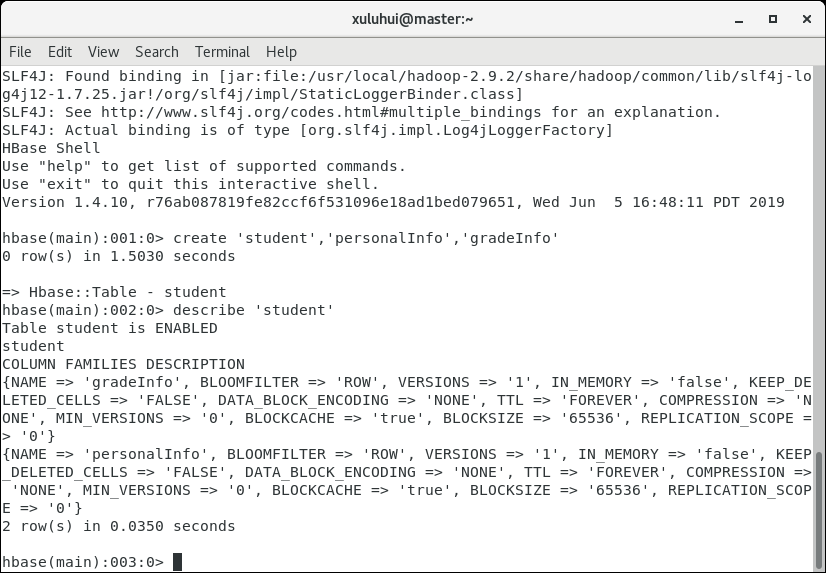


图5-30 创建表create和查看表结构describe命令

（3）使用put命令向student表中插入数据，行键值为“190809011001”，personalInfo:Sname为“xuluhui”，gradeInfo:BigData为“100”。使用命令及执行结果如图5-31所示。

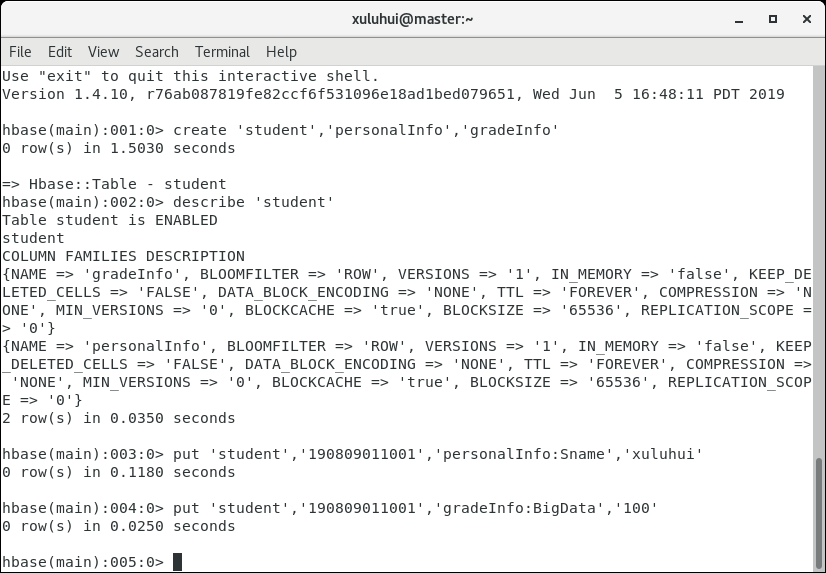


图5-31 使用put命令向表中插入数据

（4）使用scan命令查看表student的全部数据。使用命令及执行结果如图5-32所示。

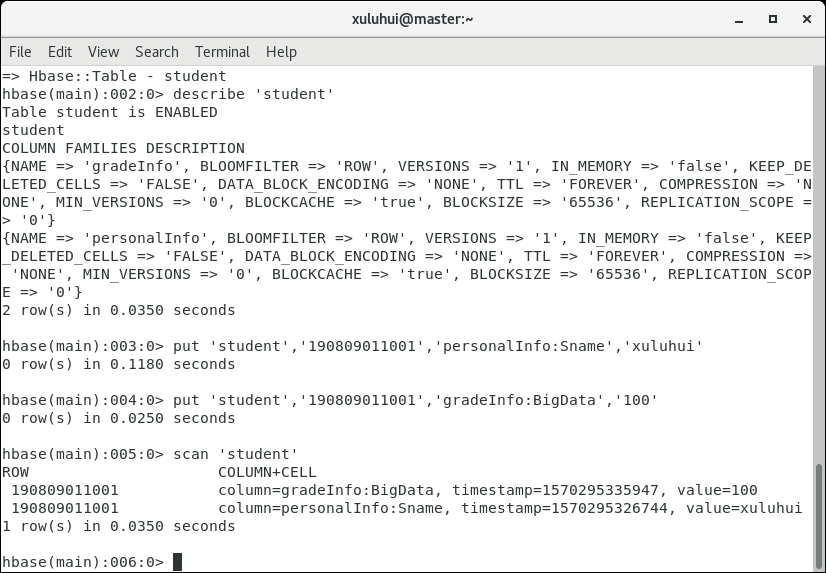


图5-32 使用scan查询表中数据

（5）为了查询表student中数据的历史版本，使用alter命令修改表结构，将列族gradeInfo的VERSIONS设置为5。使用命令及执行结果如图5-33所示，从图5-33中可以看到Describe命令的执行结果中列族gradeInfo的VERSIONS已为5。

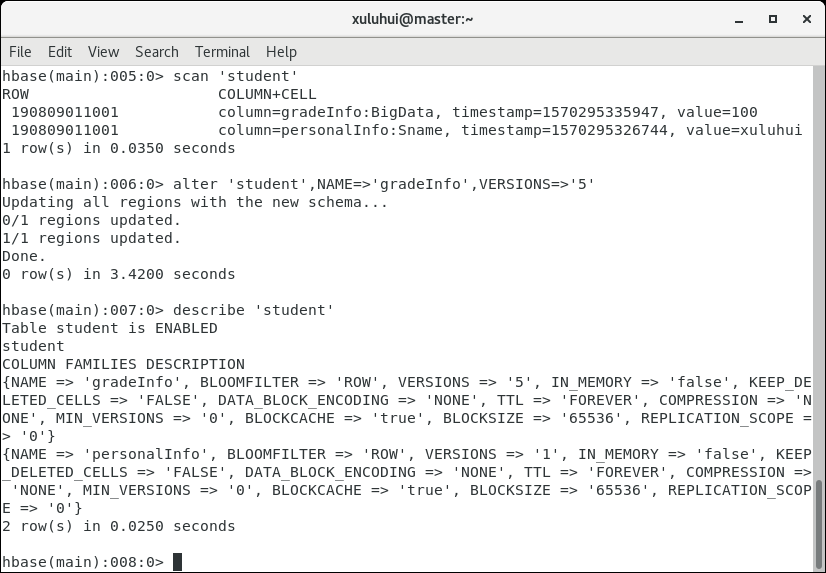


图5-33 使用alter命令修改表结构

（6）使用put命令再次向student表中插入数据，将行键值为“190809011001”的gradeInfo:BigData设置为“95”，接着第3次将行键值为“190809011001”的gradeInfo:BigData设置为“90”，最后使用get命令获取列gradeInfo:BigData的多版本数据。使用命令及执行结果如图5-34所示。这里需要注意的是，get命令中历史版本参数VERSIONS的取值“5”不加单引号，否则出错。

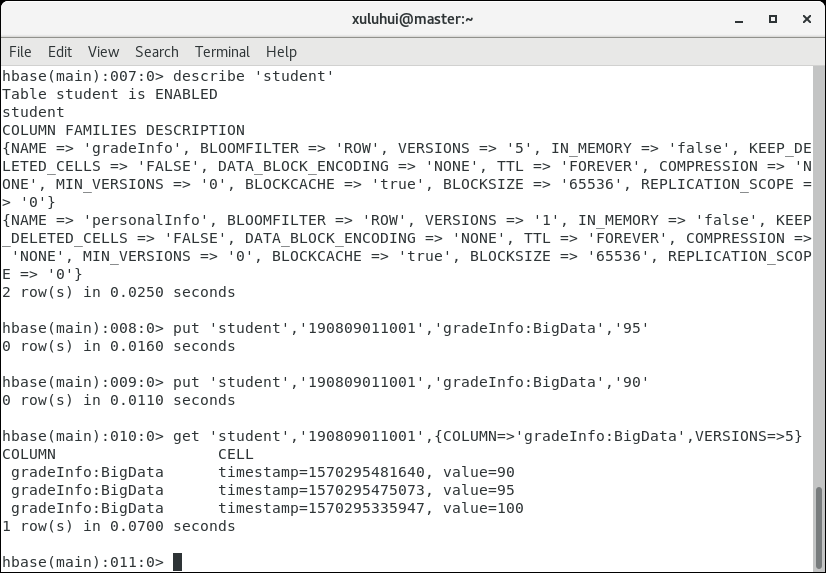


图5-34 使用get命令获取列的多版本数据

（7）打开HBase主节点的Web UI，可以看到已建立的student表，如图5-35所示。

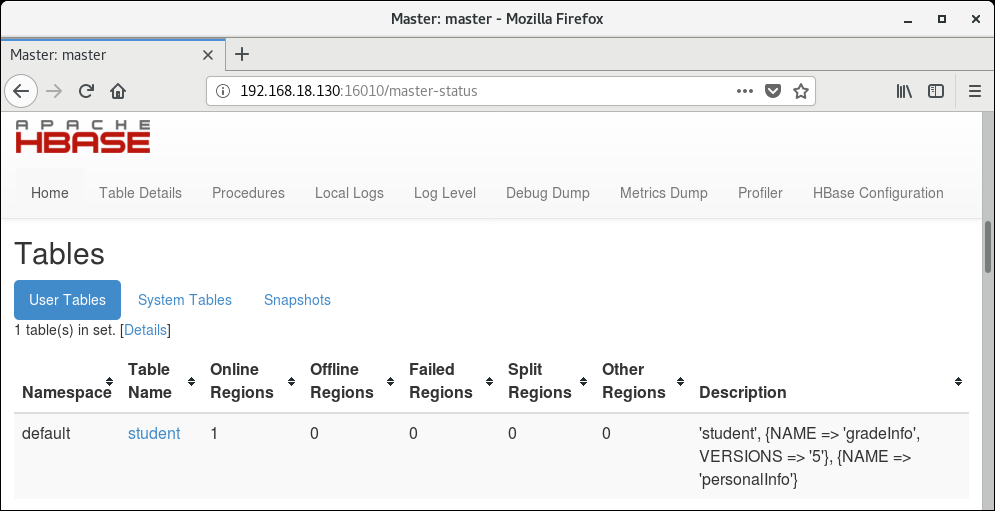


图5-35 从HBase集群主节点的Web UI界面上查看student表

（8）使用命令“zkCli.sh -server master:2181,slave1:2181,slave2:2181”连接ZooKeeper客户端，从ZooKeeper的存储树中也可以查看到建立的student表，如图5-36所示。

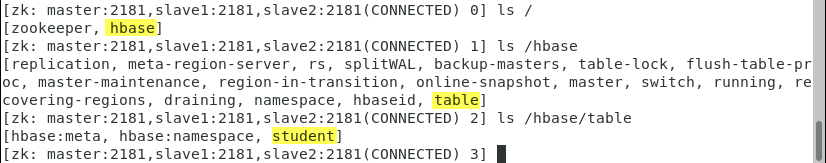


图5-36 从ZooKeeper存储树中可查看到student表

（9）由于HBase底层存储采用HDFS，所以打开HDFS Web UI，也可以查看到建立的student表，如图5-37所示。

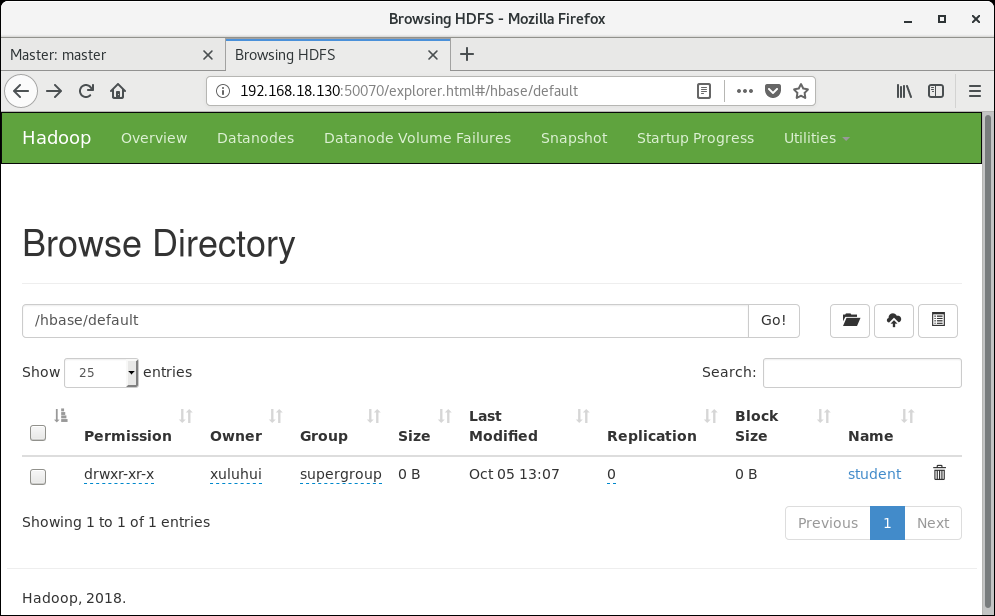


图5-37 从底层存储HDFS Web UI界面上可查看到student表

（10）删除表student。这里需要注意，不能直接使用命令“drop”删除表，表在删除之前必须处于“disable”状态，所以删除表首先要使用命令“disable”使表不可用，然后再使用命令“drop”进行删除，使用命令及执行结果如图5-38所示。

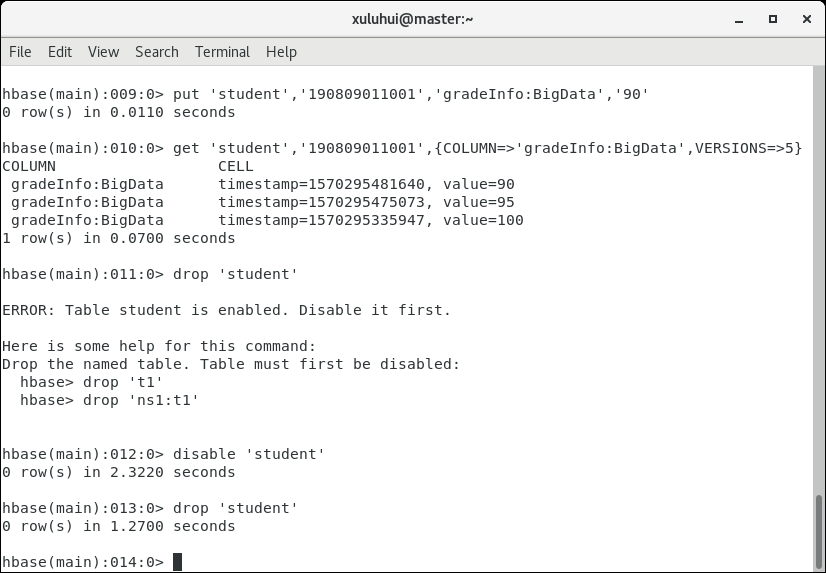
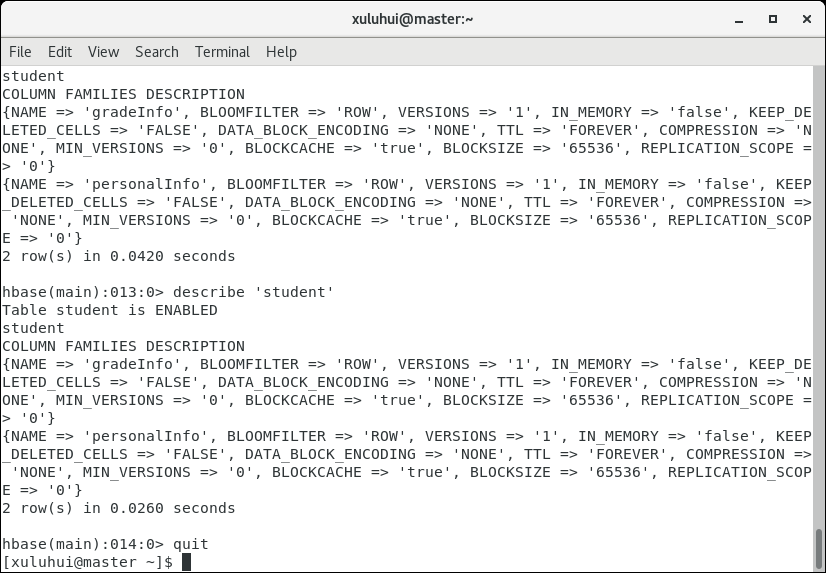


图5-38 使用disable+drop命令删除表

（11）退出HBase Shell。使用命令“quit”退出HBase Shell，使用命令及执行结果如图5-39所示。

图5-39 使用quit命令退出HBase Shell

（六）关闭全分布模式HBase集群

使用命令“stop-hbase.sh”关闭HBase集群，使用命令及执行结果如图5-40所示。

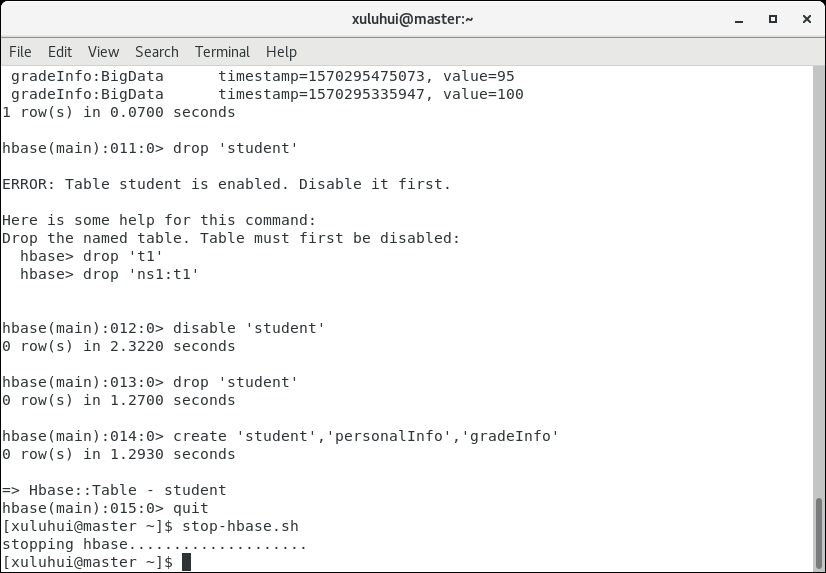


图5-40 关闭HBase集群

按照本实验设置，关闭HBase集群后HBase主节点master上的主进程HMaster、HBase从节点slave1、slave2上的从进程HRegionServer消失，同时HBase主从节点上所有与HBase相关的ZooKeeper节点文件\*.znode和进程号文件\*.pid也依次消失。

六、实验报告要求

实验报告以电子版和打印版双重形式提交。

实验报告主要内容包括实验名称、实验类型、实验地点、学时、实验环境、实验原理、实验步骤、实验结果、总结与思考等。实验报告格式如表1-9所示。

七、拓展训练

本节中，编者将通过一些简单实例来介绍如何使用HBase API编写应用程序，若要深入学习HBase编程，读者可以访问HBase官方网站提供的完整HBase API文档。

为了提高程序编写和调试效率，本书采用Eclipse工具编写Java程序，采用版本为适用于64位Linux操作系统的Eclipse IDE 2018-09 for Java Developers，关于如何在Linux下安装Eclipse，具体过程如下所示。

（一）搭建HBase开发环境Eclipse

在HBase集群的主节点上搭建HBase开发环境Eclipse，具体过程请读者参考实验项目2，此处不再赘述。

（二）HBase编程实践：HBase表的增删改

【案例5-2】使用HBase Java API实现对HBase表的增删改。

创建Java项目“HBaseExample”，在其下创建包“com.xijing.hbase”，在该包中新建类“TableCMD”，具体新建过程与前面章节相同，此处不再赘述。实现对HBase表的增删改操作的完整代码如下所示。

package com.xijing.hbase;

import java.io.IOException;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.hbase.HBaseConfiguration;

import org.apache.hadoop.hbase.HColumnDescriptor;

import org.apache.hadoop.hbase.HConstants;

import org.apache.hadoop.hbase.HTableDescriptor;

import org.apache.hadoop.hbase.TableName;

import org.apache.hadoop.hbase.client.Admin;

import org.apache.hadoop.hbase.client.Connection;

import org.apache.hadoop.hbase.client.ConnectionFactory;

import org.apache.hadoop.hbase.io.compress.Compression.Algorithm;

public class TableCMD {

private static final String TABLE\_NAME = "MY\_TABLE\_NAME\_TOO";

private static final String CF\_DEFAULT = "DEFAULT\_COLUMN\_FAMILY";

public static void createOrOverwrite(Admin admin, HTableDescriptor table) throws IOException {

if (admin.tableExists(table.getTableName())) {

admin.disableTable(table.getTableName());

admin.deleteTable(table.getTableName());

}

admin.createTable(table);

}

public static void createSchemaTables(Configuration config) throws IOException {

try (Connection connection = ConnectionFactory.createConnection(config);

Admin admin = connection.getAdmin()) {

HTableDescriptor table = new HTableDescriptor(TableName.valueOf(TABLE\_NAME));

table.addFamily(new HColumnDescriptor(CF\_DEFAULT).setCompressionType(Algorithm.NONE));

System.out.print("Creating table. ");

createOrOverwrite(admin, table);

System.out.println(" Done.");

}

}

public static void modifySchema (Configuration config) throws IOException {

try (Connection connection = ConnectionFactory.createConnection(config);

Admin admin = connection.getAdmin()) {

TableName tableName = TableName.valueOf(TABLE\_NAME);

if (!admin.tableExists(tableName)) {

System.out.println("Table does not exist.");

System.exit(-1);

}

HTableDescriptor table = admin.getTableDescriptor(tableName);

// Update existing table

HColumnDescriptor newColumn = new HColumnDescriptor("NEWCF");

newColumn.setCompactionCompressionType(Algorithm.GZ);

newColumn.setMaxVersions(HConstants.ALL\_VERSIONS);

admin.addColumn(tableName, newColumn);

// Update existing column family

HColumnDescriptor existingColumn = new HColumnDescriptor(CF\_DEFAULT);

existingColumn.setCompactionCompressionType(Algorithm.GZ);

existingColumn.setMaxVersions(HConstants.ALL\_VERSIONS);

table.modifyFamily(existingColumn);

admin.modifyTable(tableName, table);

// Disable an existing table

admin.disableTable(tableName);

// Delete an existing column family

admin.deleteColumn(tableName, CF\_DEFAULT.getBytes("UTF-8"));

// Delete a table (Need to be disabled first)

admin.deleteTable(tableName);

}

}

public static void main(String[] args) throws IOException {

Configuration config = HBaseConfiguration.create();

//Add any necessary configuration files (hbase-site.xml, core-site.xml)

config.addResource(new Path(System.getenv("HBASE\_CONF\_DIR"), "hbase-site.xml"));

config.addResource(new Path(System.getenv("HADOOP\_CONF\_DIR"), "core-site.xml"));

createSchemaTables(config);

modifySchema(config);

}

}

思考与练习题

使用HBase Java API编程程序，实现新建HBase表、插入数据的功能，打成JAR包并提交集群执行，观察运行结果。

参考文献

[1] GEORGE L. HBase权威指南[M]. 代志远,译. 北京:人民邮电出版社,2013.

[2] CHANG F, DEAN J, GHEMAWAT S, et al. Bigtable: A Distributed Storage System for Structured Data[C]// Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation (OSDI’06), USENIX, 2006:205-218.

[3] Apache HBase[EB/OL]. https://hbase.apache.org/.

[4] GitHub-Apache HBase[EB/OL]. https://github.com/apache/hbase.

[5] Apache Software Foundation. Apache HBase Download[EB/OL]. http://hbase.apache.org/downloads.html.

[6] Apache Software Foundation. Apache HBase Reference Guide[EB/OL]. https://hbase.apache.org/book.html.

[7] Apache Software Foundation. Apache HBase API[EB/OL]. https://hbase.apache.org/apidocs/index.html.

[8] Carol McDonald. An In-Depth Look at the HBase Architecture[EB/OL]. [2015-8-7]. https://mapr.com/blog/in-depth-look-hbase-architecture/.