第5章 HBase分布式数据库

学习目标：

- 了解 HBase 的基础知识，能够说出 HBase 的特点和数据模型。
- 熟悉 HBase 架构，能够叙述 HBase 中各组件的作用。
- 了解物理存储，能够说出 HBase 如何存储数据。
- 熟悉 HBase 读写数据流程，能够叙述 HBase 读写数据的流程。
- 掌握 HBase 高可用集群的搭建，能够独立完成 HBase 高可用集群的搭建。
- 掌握 HBase 的 Shell 操作，能够使用 HBase Shell 操作 HBase。
- 掌握 HBase 的 Java API 操作，能够使用 Java API 操作 HBase。
- 掌握 HBase 集成 Hive，能够实现通过 Hive 向 HBase 的数据表插入数据。

在分布式计算环境下，Spark 可以将处理后的数据实时写入 HBase 数据库，以满足对大规模数据存储和快速访问的需求。HBase 是一种面向列的分布式数据库，专为处理海量数据而设计。与传统的行式数据库（如 MySQL 和 Oracle）不同，HBase 的列式存储允许灵活地添加新的列，从而轻松适应不断变化的数据结构。这种特性使得 Spark 能够将实时计算结果高效地存储到 HBase 中。本章详细讲解 HBase 分布式数据库的相关知识。

5.1 HBase 的基础知识

5.1.1 HBase 的简介

“沉淀”往往是通过对技术实践和经验的总结和提炼，形成深刻的认识和经验，从而提高技术水平和解决实际问题的能力。HBase 起源于 Google 公司发表的 BigTable 论文，它是一个高可靠性、高性能、面向列、可扩展的分布式数据库。HBase 的目标是存储并处理大型的数据，更具体来说是仅需使用普通的硬件配置，就能够处理由成千上万的行和列所组成的大型数据。HBase 具有如下的显著特点。

（1）容量大。HBase 中的表可以存储成千上万的行和列组成的数据。

（2）面向列。HBase 是面向列的存储，支持独立检索。面向列的存储是指其数据在表中是按照某列存储的，根据数据动态地增加列，并且可以单独对列进行各种操作。

（3）多版本。HBase 中表的每个列的数据存储都有多个版本。例如，存储个人信息的 HBase 数据表中，如果某个人多次更换过家庭住址，那么记录家庭住址的数据就会有多个版本。

(4) 稀疏性。由于 HBase 中表的列允许为空,并且空列不会占用存储空间,所以表可以设计得非常稀疏。

(5) 扩展性。HBase 的底层依赖于 HDFS。当磁盘空间不足时,可以动态地增加服务器,即增加 DataNode 节点,以增加磁盘空间,从而避免像 MySQL 数据库那样,进行数据迁移。

(6) 高可靠性。由于 HBase 底层使用的是 HDFS,而 HDFS 本身会备份数据,所以在 HBase 出现宕机时,HDFS 能够保证数据不会发生丢失或损坏。

5.1.2 HBase 的数据模型

HBase 的数据存储在行列式的表格中,是一个多维度的映射模型,其数据模型如图 5-1 所示。

Row Key	Timestamp	Column Family:c1		Column Family:c2		Column Family:c3	
		Column	Value	Column	Value	Column	Value
r1	t7	c1:col-1	value-1			c3:col-1	value-1
	t6	c1:col-2	value-2			c3:col-2	value-1
	t5	c1:col-3	value-3				
	t4						
r2	t3	c1:col-1	value-1	c2:col-1	value-1	c3:col-1	value-1
	t2	c1:col-2	value-2				
	t1	c1:col-3	value-3				

图 5-1 HBase 的数据模型

从图 5-1 可以看出,图中包含了很多的字段,这些字段分别表示不同的含义,具体介绍如下。

1. Row Key(行键)

Row Key 表示行键,是 HBase 数据表中的每行数据的唯一标识符。在 HBase 中,Row Key 按照字典顺序进行存储,因此,设计一个好的 Row Key 对于数据的存储和检索至关重要。Row Key 是检索数据的主要方式之一,通过设计高效的 Row Key,可以更快地检索到所需的数据,避免全表扫描或不必要的数据查找。此外,良好的 Row Key 设计还可以确保相关的数据存储在一起,从而减少磁盘寻址时间,提高检索速度。

2. Timestamp(时间戳)

Timestamp 表示时间戳,记录每次操作数据的时间,通常作为数据的版本号。

3. Column(列)

列由列族和列标识两部分组成,两者之间用“:”分隔。例如在列族 info 中,通过列标识 name 标识的列为 info:name。创建 HBase 数据表时不需要指定列,因为列是可变的,非常灵活。

4. Column Family(列族)

在 HBase 中,列族由多个列组成。在同一个表里,不同列族有不同的属性,但是同一个列族内的所有列都会有相同的属性,因为属性定义在列族级别上。在图 5-1 中,c1,c2,c3 均为列族名。

5.2　深入学习 HBase 原理

在使用 HBase 之前，学习 HBase 原理可以更好地理解 HBase。接下来，本节从 HBase 架构、物理存储以及 HBase 读写数据流程详细讲解 HBase 原理。

5.2.1　HBase 架构

HBase 构建在 Hadoop 之上，Hadoop 中的 HDFS 为 HBase 提供了高可靠的底层存储支持，同时 Hadoop 中的 MapReduce 为 HBase 提供了高性能的计算能力，而 ZooKeeper 为 HBase 提供了稳定服务和容错机制。下面通过图 5-2 介绍 HBase 架构。

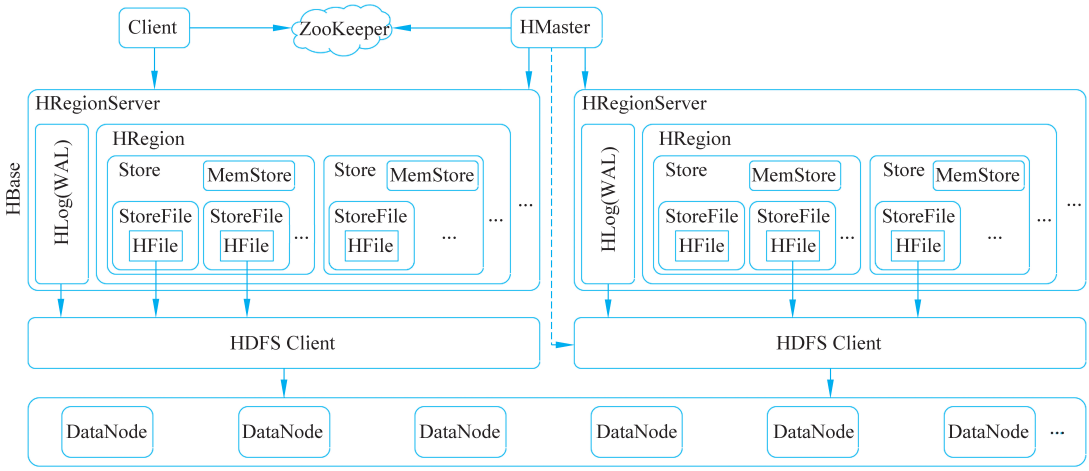

图 5-2　HBase 架构

从图 5-2 可以看出，HBase 包含多个组件。下面针对 HBase 架构中的组件进行详细介绍。

（1）Client。即用户提交相关命令操作 HBase 的客户端，它通过 RPC 协议与 HBase 进行通信。

（2）ZooKeeper。即分布式协调服务，在 HBase 集群中的主要作用是监控 HRegionServer 的状态，并将 HRegionServer 的状态实时通知给 HMaster，确保集群中只有一个 HMaster 在工作。

（3）HMaster。即 HBase 集群的主节点，用于协调多个 HRegionServer，主要用于监控 HRegionServer 的状态以及平衡 HRegionServer 之间的负载。除此之外，HMaster 还负责为 HRegionServer 分配 HRegion。

在 HBase 中，如果有多个 HMaster 节点共存，提供服务的只有一个 HMaster，其他的 HMaster 处于待命的状态。如果当前提供服务的 HMaster 节点宕机，那么其他的 HMaster 通过 ZooKeeper 选举出一个激活的 HMaster 节点接管 HBase 的集群。

（4）HRegionServer。即 HBase 集群的从节点，它包括了多个 HRegion，主要用于响应用户的 I/O 请求，并与 HDFS 交互进行读写数据。

（5）HRegion。即 HBase 数据表的分片，每个 HRegion 中保存的是 HBase 数据表中某

段连续的数据。

(6) Store。每个 HRegion 包含一个或多个 Store。每个 Store 用于管理一个 HRegion 上的列族。

(7) MemStore。即内存级缓存,MemStore 存放在 Store 中,用于保存修改的键值对(Key,Values)形式的数据。当 MemStore 存储的数据达到一个阈值时,默认为 128MB,数据就会被执行刷写操作,将数据写入 StoreFile 文件。MemStore 的刷写操作是由专门的线程负责的。

(8) StoreFile。MemStore 中的数据写到文件后就是 StoreFile,StoreFile 底层是以 HFile 的格式保存在 HDFS 上。

(9) HLog(WAL)。即预写日志文件,负责记录 HBase 的修改。当 HBase 读写数据时,数据首先会被写入 HLog,然后再写入内存。这样,即使在写入内存之前出现故障,数据仍然可以通过 HLog 进行恢复。

5.2.2 物理存储

HBase 最重要的功能就是存储数据,下面从 4 方面详细介绍 HBase 的物理存储。

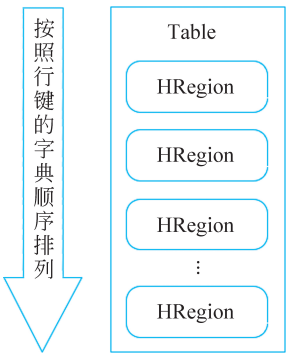

图 5-3 HBase 数据表中数据的存储方式

(1) HBase 数据表的数据按照行键的字典顺序进行排列。此外,数据还被切分多个 HRegion 存储,每个 HRegion 存储一段连续的行键范围。存储方式如图 5-3 所示。

(2) 多个 HRegion 在一个 HRegionServer 上存储。一个 HRegionServer 上可以存储多个 HRegion,但是每个 HRegion 只能被分布到一个 HRegionServer 上,这种设计可以确保 HBase 的数据在 HRegionServer 之间进行均衡分布,分布方式如图 5-4 所示。

(3) 动态切分 HRegion。每个 HRegion 存储的数据是有限的,当一个 HRegion 增大到一定的阈值时,会被等切分成两个新的 HRegion,保证数据均匀分布和存储的可扩展性,切分方式如图 5-5 所示。

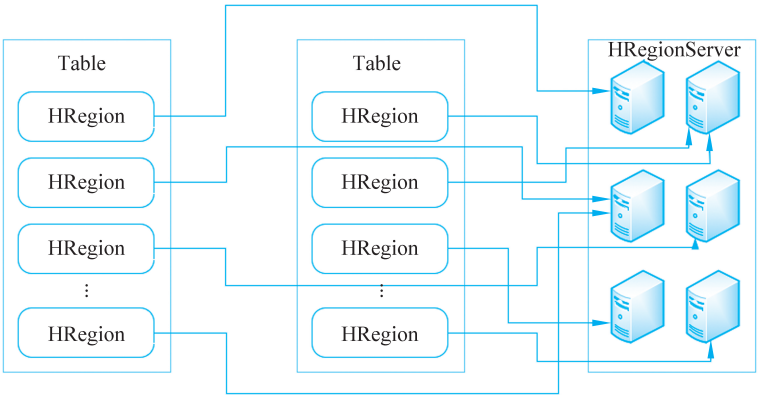

图 5-4 HRegion 的分布方式

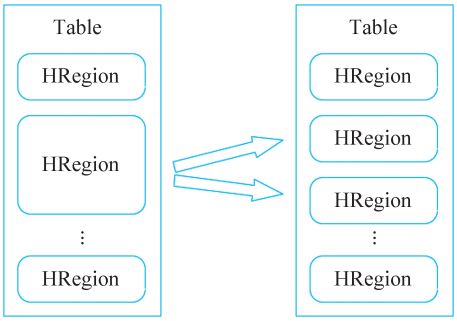

图 5-5　HRegion 的切分方式

(4) 写入数据到 MemStore 和刷写到 StoreFile。MemStore 中存储的是用户写入的数据，一旦 MemStore 存储达到一定的阈值(默认为 128MB)时，数据就会被刷写到新生成的 StoreFile 中(底层是 HFile)，该文件是以 HFile 的格式存储到 HDFS 上，具体如图 5-6 所示。

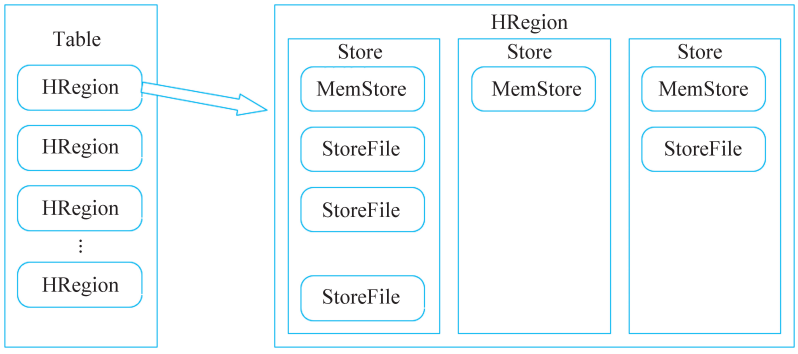

图 5-6　HBase 数据表的存储

5.2.3　HBase 读写数据流程

HBase 读写数据需要依靠 ZooKeeper 来实现，这是因为 ZooKeeper 中存储了 HBase 中 ROOT 表的位置信息，而 ROOT 表又存储了 META 表的 HRegion 信息以及所有 HRegionServer 的地址。

下面介绍 HBase 读写数据的流程。

1. HBase 写数据流程

HBase 写数据是指 Client 向 HBase 的数据表写入数据，下面通过图 5-7 来了解 HBase 写数据流程。

从图 5-7 可以看出，HBase 写数据流程大概分为 7 个步骤，具体流程如下。

(1) Client 向 ZooKeeper 发送请求，获取 META 表所在 HRegionServer 的地址信息。

(2) ZooKeeper 将 META 表所在 HRegionServer 的地址信息返回给 Client。

(3) Client 访问对应的 HRegionServer，获取 META 表记录的元数据，从而找到表对应的所有 HRegion，并根据 Region 存储数据的范围确定具体写入的目标 HRegion。

(4) HRegionServer 将 META 表记录的元数据信息以及目标 HRegion 的信息返回给

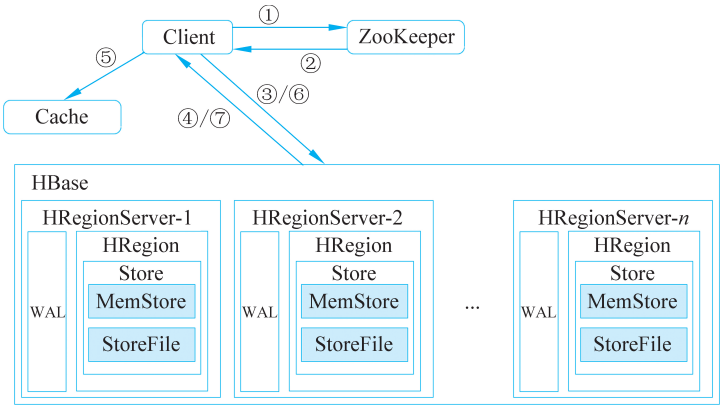

图 5-7 HBase 写数据流程

Client。

(5) Client 将 META 表记录的元数据信息以及目标 HRegion 缓存到 Client 的 Cache 中,方便下次写数据时可以直接访问。

(6) Client 向 HRegionServer 发送数据,HRegionServer 将得到的数据暂时存储在 WAL 中,如果 MemStore 存储的数据达到刷写操作的阈值时,数据将被刷写到 StoreFile 中。

(7) 一旦数据成功存储到 StoreFile 中,HRegionServer 将存储完成的信息返回给 Client,完成数据写入过程。

2. HBase 读数据流程

HBase 读数据流程与写数据流程类似,下面通过图 5-8 来了解 HBase 读数据流程。

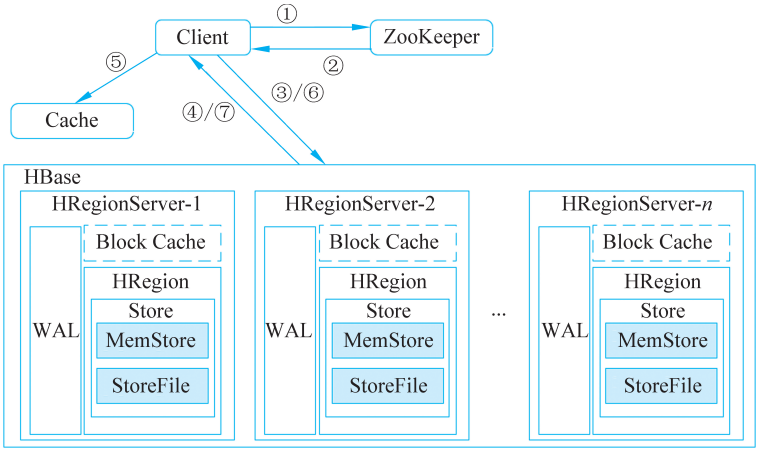

图 5-8 HBase 读数据流程

从图 5-8 可以看出,HBase 读数据流程大概分为 7 个步骤,与写数据流程不同的是,读数据流程中的 HRegionServer 中多了 Block Cache,Block Cache 的作用是读缓存,即将读取的数据缓存到内存中,提高读取数据的效率。HBase 读数据具体流程如下。

(1) Client 向 ZooKeeper 发送请求,获取 HBase 中 META 表所在 HRegionServer 的地址信息。

（2）ZooKeeper 将 META 表所在 HRegionServer 的地址信息返回给 Client。

（3）Client 访问对应的 HRegionServer，获取 META 表记录的元数据，从而找到表对应的所有 HRegion，并根据 Region 存储数据的范围确定要读取的目标 HRegion。

（4）HRegionServer 将 META 表记录的元数据信息以及 HRegion 返回给 Client。

（5）Client 将接收到的 META 表记录的元数据信息以及 HRegion 缓存到 Client 的 Cache 中，方便下次读数据时访问。

（6）Client 向 HRegionServer 请求读取数据，HRegionServer 在 Block Cache、MemStore 和 Store File 中查询目标数据，并将查到的所有数据进行合并。

（7）HRegionServer 将合并后的最终结果返回给 Client。

【提示】 ROOT 表是 HBase 中的一个特殊表，它存储了 META 表的位置信息。在 HBase 中，META 表是一个描述表和 HRegion 分布的元数据表。META 表中的每一行记录都代表一个表或 HRegion 的元数据信息，包括表名、列族以及负责该 HRegion 的 HRegionServer 的地址等。

5.3　搭建 HBase 高可用集群

在普通的 HBase 集群中会存在单点故障问题，例如，当主节点发生宕机时，整个集群将无法正常工作，针对这样的问题，可以利用 ZooKeeper 提供的选举机制部署一个高可用的 HBase 集群来解决。这样即使主节点宕机，其他节点仍然可以正常工作，保证集群的稳定性。

下面以虚拟机 Hadoop1、Hadoop2 和 Hadoop3 为例讲解如何搭建 HBase 高可用集群。HBase 高可用集群的规划方式如图 5-9 所示。

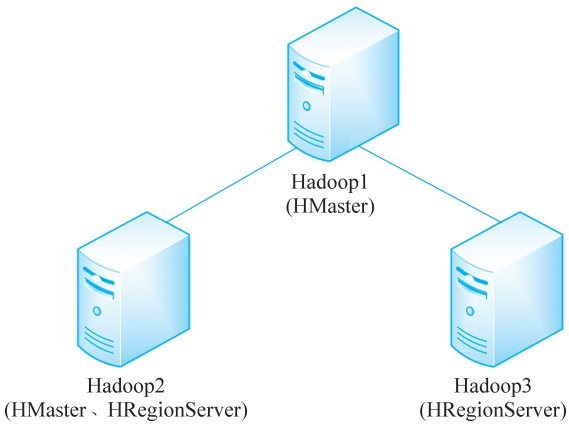

图 5-9　HBase 高可用集群的规划

从图 5-9 可以看出，HBase 高可用集群中虚拟机 Hadoop1 和 Hadoop2 是主节点，虚拟机 Hadoop2 和 Hadoop3 是从节点。这里之所以将虚拟机 Hadoop2 既部署为主节点也部署为从节点，目的是避免 HBase 集群主节点宕机导致的单点故障问题，同时也为了提高 HBase 集群读写数据的效率。

接下来，分步骤讲解如何搭建 HBase 高可用集群，具体步骤如下。

1. 下载 HBase 安装包

本书使用的 HBase 版本为 2.4.9,通过访问 HBase 官网,下载 HBase 安装包 hbase-2.4.9-bin.tar.gz。

2. 上传 HBase 安装包

在虚拟机 Hadoop1 的/export/software 目录执行 rz 命令,将下载好的 HBase 安装包上传到虚拟机的/export/software 目录。

3. 安装 HBase

通过对 HBase 安装包进行解压操作安装 HBase,将 HBase 安装到存放安装程序的目录/export/servers,在/export/software 目录执行如下命令。

```
$ tar -zxvf hbase-2.4.9-bin.tar.gz -C /export/servers/
```

4. 配置 HBase 环境变量

分别在虚拟机 Hadoop1、Hadoop2 和 Hadoop3 执行“vi /etc/profile”命令编辑系统环境变量文件 profile,在该文件的尾部添加如下内容。

```
export HBASE_HOME=/export/servers/hbase-2.4.9
export PATH=$PATH:$HBASE_HOME/bin
```

成功配置 HBase 环境变量后,保存并退出系统环境变量文件 profile 即可。不过此时在系统环境变量文件中添加的内容尚未生效,还需要分别在虚拟机 Hadoop1、Hadoop2 和 Hadoop3 执行“source /etc/profile”命令初始化系统环境变量,使配置的 HBase 环境变量生效。

5. 修改配置文件

为了确保 HBase 高可用集群能够正常启动,必须对 HBase 的配置文件进行相关的配置,具体步骤如下。

(1) 在虚拟机 Hadoop1 中执行“cd /export/servers/hbase-2.4.9/conf”命令进入 HBase 安装目录的 conf 目录,在该目录下执行“vi hbase-env.sh”命令编辑 hbase-env.sh 配置文件,在 hbase-env.sh 配置文件底部添加如下内容。

```
export HBASE_DISABLE_HADOOP_CLASSPATH_LOOKUP="true"
export JAVA_HOME=/export/servers/jdk1.8.0_241
export HBASE_MANAGES_ZK=false
```

上述内容添加完成后,保存并退出 hbase-env.sh 配置文件。关于上述内容的具体介绍如下。

- HBASE_DISABLE_HADOOP_CLASSPATH_LOOKUP:用于设置 HBase 在运行时是否自动查找 Hadoop 类路径,设置为 true 表示 HBase 在运行时不自动查找 Hadoop 类路径。
- JAVA_HOME:用于指定 HBase 使用的 JDK,这里使用的是本地安装的 JDK。
- HBASE_MANAGES_ZK:用于指定 HBase 高可用集群主节点选举机制,设置为 false 表示使用的是本地安装的 ZooKeeper 集群。

(2) 在 HBase 安装目录的 conf 目录执行 vi hbase-site.xml 命令编辑 hbase-site.xml 配置文件,将 hbase-site.xml 配置文件的<configuration>标签中的默认内容修改为如下

内容。

```
<property>
      <name>hbase.rootdir</name>
      <value>hdfs://hadoop1:9000/hbase</value>
</property>
<property>
    <name>hbase.cluster.distributed</name>
    <value>true</value>
</property>
<property>
    <name>hbase.zookeeper.quorum</name>
    <value>hadoop1:2181,hadoop2:2181,hadoop3:2181</value>
</property>
```

上述内容修改完成后，保存并退出 hbase-site.xml 配置文件。关于上述参数的介绍具体如下。

- 参数 hbase.rootdir 用于指定 HBase 集群在 HDFS 上存储的路径。
- 参数 hbase.cluster.distributed 用于指定 HBase 集群是否为分布式的，设置为 true 表示指定 HBase 集群是分布式的。
- 参数 hbase.zookeeper.quorum 用于指定 ZooKeeper 集群中所有 ZooKeeper 服务的地址。

(3) 在 HBase 安装目录的 conf 目录执行 vi regionservers 命令编辑 regionservers 配置文件，将 regionservers 配置文件中的默认内容修改为如下内容。

```
hadoop2
hadoop3
```

上述内容表示在主机名为 hadoop2 和 hadoop3 的虚拟机 Hadoop2 和 Hadoop3 中运行 HRegionServer。上述内容修改完成后，保存并退出 regionservers 配置文件。

(4) 在 HBase 安装目录的 conf 目录执行 vi backup-masters 命令编辑 backup-masters 配置文件，在 backup-masters 配置文件中添加如下内容。

```
hadoop2
```

上述内容表示在主机名为 hadoop2 的虚拟机 Hadoop2 中运行备用的 HMaster。上述内容添加完成后，保存并退出 backup-masters 配置文件。

6. 分发 HBase 安装目录

执行 scp 命令将虚拟机 Hadoop1 的 HBase 安装目录分发至虚拟机 Hadoop2 和 Hadoop3 中存放安装程序的目录，具体命令如下。

```
#将 HBase 安装目录分发至虚拟机 Hadoop2 中存放安装程序的目录
$ scp -r /export/servers/hbase-2.4.9/ hadoop2:/export/servers/
#将 HBase 安装目录分发至虚拟机 Hadoop3 中存放安装程序的目录
$ scp -r /export/servers/hbase-2.4.9/ hadoop3:/export/servers/
```

7. 启动 ZooKeeper 和 Hadoop

在启动 HBase 高可用集群之前，需要先启动 ZooKeeper 服务和 Hadoop 集群，具体命令如下。

```
#在虚拟机 Hadoop1、Hadoop2 和 Hadoop3 上分别启动 ZooKeeper 服务
$ zkServer.sh start
#在虚拟机 Hadoop1 上执行启动 Hadoop 集群的命令
$ start-all.sh
```

8. 启动 HBase

在虚拟机 Hadoop1 上执行如下命令启动 HBase 集群。

```
$ start-hbase.sh
```

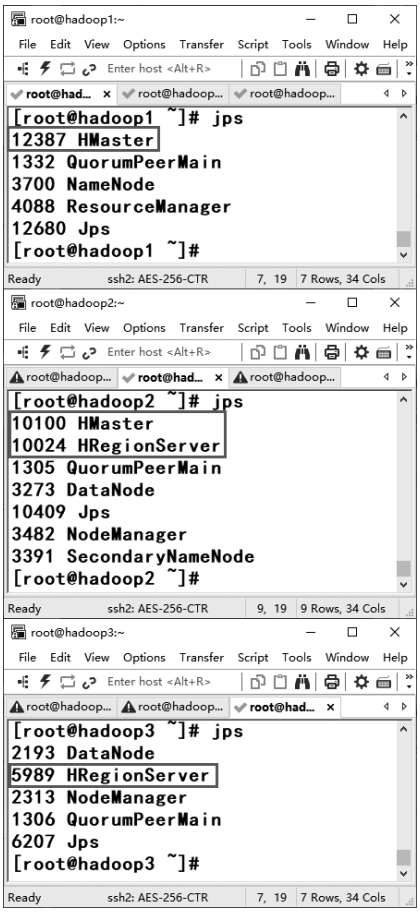

图 5-10 查看 HBase 运行状态

如果要关闭 HBase 高可用集群,则在虚拟机 Hadoop1 执行“stop-hbase.sh”命令即可。

9. 查看 HBase 运行状态

分别在虚拟机 Hadoop1、Hadoop2 和 Hadoop3 执行 jps 命令查看 HBase 运行状态,如图 5-10 所示。

从图 5-10 可以看出,虚拟机 Hadoop1 运行着 HBase 的 HMaster;虚拟机 Hadoop2 运行着 HBase 的 HMaster 和 HRegionServer;虚拟机 Hadoop3 运行着 HBase 的 HRegionServer,说明 HBase 启动成功。

10. 通过 Web UI 查看 HBase 运行状态

HBase 默认提供了 16010 端口用于通过 Web UI 查看 HBase 运行状态。分别在本地计算机的浏览器输入 http://hadoop1:16010 和 http://hadoop2:16010 查看 HBase 集群运行状态,以及备用 HMaster 的运行状态,如图 5-11 和图 5-12 所示。

从图 5-11 可以看出,虚拟机 Hadoop1 是 HBase 的主节点,虚拟机 Hadoop2 和 Hadoop3 是 HBase 的从节点。

从图 5-12 可以看出,虚拟机 Hadoop2 运行着备用 HMaster,并且显示了当前激活状态的 HMaster 运行在虚拟机 Hadoop1。

HBase 高可用集群的搭建相对简单,但在搭建 HBase 高可用集群时仍然需要明白以细心严谨的态度对待这一过程的重要性。这不仅有助于顺利完成 HBase 高可用集群的搭建,还能培养我们严谨的思维和端正的态度,为综合发展打下坚实的基础。

多学一招:修改日志信息输出级别

默认情况下,HBase 日志信息输出级别为 INFO,这种情况下会输出大量冗余信息,为了提高查看 HBase 运行时信息的便利性,建议将日志信息输出级别修改为 ERROR,这样可以减少不必要的输出,使日志信息更加简洁明了。

在虚拟机 Hadoop1 的 HBase 安装目录的 conf 目录执行 vi log4j.properties 命令编辑