第四十九章：Hbase分布式数据库

**一、Hbase数据库概述；**

**二、Hbase体系结构；**

**三、Hbase数据库模型；**

**四、总结Hbase整体特点；**

**五、案例：搭建Hbase分布式数据库系统；**

**一、Hbase数据库概述；**

**概述：**Hbase是一个基于HDFS的面向列的分布式数据库，源于Google的BigTable基于GFS进行分布式数据存储一样，前文提到，Hbase是基于流式数据访问，对于第时间延迟的数据访问并不适合在HDFS上运行，所以需要实时性的随机访问超大规模的数据集，使用Hbase则是更好的选择；



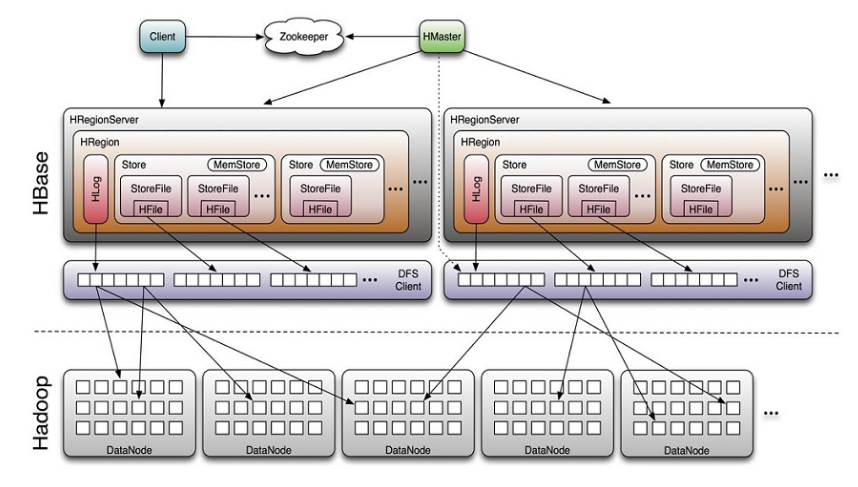
**作用：**Hbase作为典型的非关系型数据库，Nosql数据库主要分为以下几类：

* 基于键值对存储的类型；
* 基于文档存储的类型；
* 基于列存储的类型；
* 基于图形数据存储的类型；

在Nosql领域中，Hbase本身不是最优秀的，但得益于与hadoop的整合，为其带来了强大的扩展空间。Hbase本质只有插入操作，更新删除等操作都是通过插入操作来完成，这是由于底层HDFS流式访问（一次写入，多次读取）决定的，每次插入数据时，数据会带有“时间戳”的标记，形成多个版本，Hbase对于一个数据会保留其固定的版本数量，如果在查询时，也是显示出距离当前时间最近的一个新版本；

**二、Hbase体系结构；**

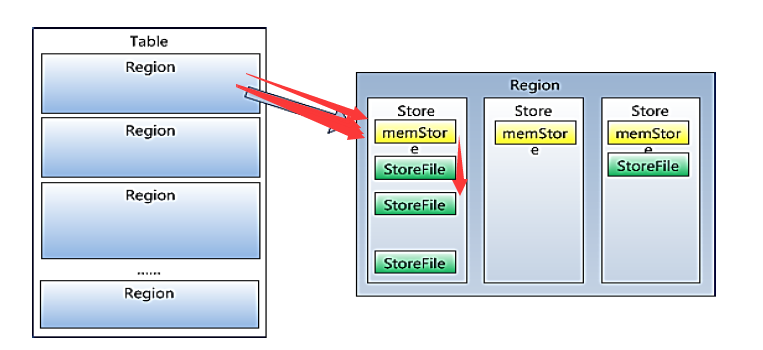
**体系结构：**



**架构分析：**Hbase体系结构由单个HMaster服务器和多个HRegion Server服务器组成，而所有这些服务器是通过ZooKeeper来进行协调并处理各个服务器运行期间可能遇见的问题；

**组件分析：**

* **HStore：**多个HStore组成一个HRegion，本身由两部分组成：Memstore和Storefile。首先用户写入的数据存放到Memstore中，当Memstore满了后刷入Storefile；
* **HRegion：**由多个HStore组成，Hbase使用表存储数据集，表由行和列组成，但与传统关系型数据库不同的是，当表的大小超过设定的值时，Hbase会自动将表划分为不同的区域HRegion（此操作也称之为HRegion分裂），它是Hbase集群上分布式存储和负载均衡的最小单位，这一点和HDFS中文件与文件块存储的概念类似；



* **Hlog：**存储数据日志，到达HRegion上的写操作首先被追加到日志中，然后才被加载到Memstore，主要功能为故障修复，当某台HRegionServer发生故障，新的HRegionServer在加载HRegion的时候可以通过Hlog对数据进行恢复；
* **HRegionServer：**由多个HRegion组成，在整个集群中可能存在多个节点，每个节点只能运行一个HRegionServer，负责对HDFS中读写数据和管理HRegion和Hlog；
* **HMaster：**每台HRegionServer都会与HMaster进行通信，HMaster的主要任务就是告诉HRegionServer它需要维护哪些HRegion，具体功能如下：

1.管理用户对表的增删改查操作；

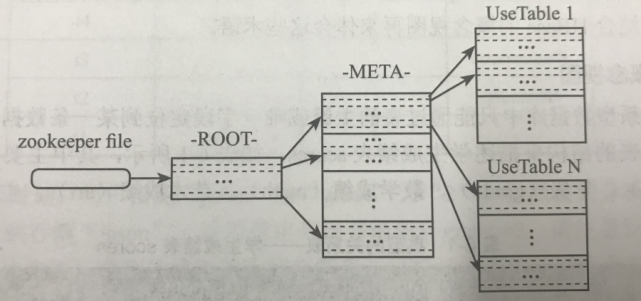
2.管理HRegionServer的负载均衡，动态调整HRegion分布；

3.在HRegion分裂后，负责新的HRegion的分配；

4.在HRegionServer停机后，负责失效HRegionServer上的HRegion的迁移；

* **ZooKeeper：**存储的是Hbase中的ROOT表（根数据表）和META表（元数据表），元数据表保存普通用户表的HRegion标识符信息， 标识符格式为：表名+开始主键+唯一ID。随着HRegion的分裂，标识符信息也会发生变化，分成多个HRegion后，需要由一个根数据表来贯穿多个元数据表；

此外，ZooKeeper还负责HRegionServer故障时，通知HMaster进行HRegion迁移；若HMaster出现故障，ZooKeeper负责恢复HMaster，并且保证有且只有一个HMaster正在运行；



* **Client：**客户端访问Hbase的单位，访问时，首先访问Zookeeper--ROOT--META--table；

**三、Hbase数据库模型；**

**1.数据模型：**

**表（table）：**不存储值为null的数据，索引是行关键字、列关键字、时间戳；

**行关键字（row key）：**行的主键，唯一标识一行数据；

**列族（Colume Family）：**行中的列被分为“列族”，同一个列族的所有成员具有相同的列族前缀，一个表的列族必须在创建表时预先定义，格式（列名：修饰符）；

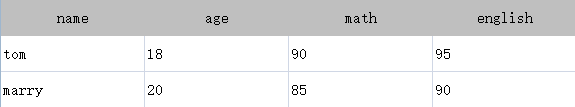
**列关键字（Colume key）：**列键，格式为：<family>:<qualifer>，family是列族名；qualifer是列族修饰符，表示列族中的一个成员；

**存储单元格（Cell）：**在Hbase中，值作为一个单元保存在单元格中，要定位一个单元，需要满足“行键+列键+时间戳”三个要素；

**时间戳（Timestamp）：**插入单元格时的时间戳，默认作为单元格的版本号；

**2.存储方式：**

**关系型数据库：**



主键设置为name列，查找时根据学生名字可以很容易的实现查找，那么请思考以下问题；

* 如果现在新增加一门课程，如何在不改变表结构的情况下进行保存新课程的成绩呢？
* 如果tom同学数学成绩参加了补考，如何记录其同学的两次数学成绩？
* 如若tom同学数学没有成绩，那么表中值为null，即使为空，也会占用存储空间；

**HBase数据库：**



在不同时间插入不同数据时，会生成时间戳，并且在列族内生成数据记录；

在HBase数据库实际存储时，其表内空值不计入存储空间内；

**四、总结Hbase整体特点：**

HBase就是这样一个基于列模式的映射数据库，它只能表示简单的键值的映射关系。与关系型数据库相比，它有如下特点：

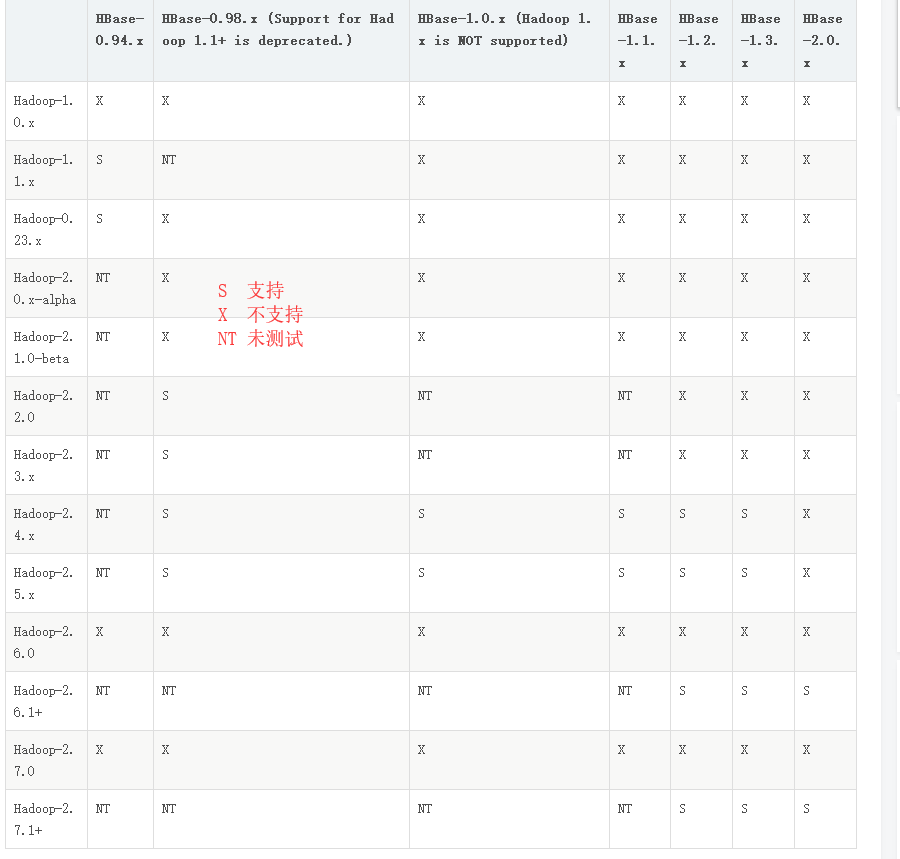
* 数据类型： HBase只有简单的字符串类型，它只保存字符串。而关系型数据库有丰富的类型选择和存储方式；
* 数据操作：HBase 只有简单的插入、查询、删除、清空等操作，表和表之间是分离的，没有复杂的表和表之间的关系，所以不能、也没有必要实现表和表之间的关联操作。而关系型数据库有多种连接操作；
* 存储模式：HBase 是基于列存储的，每个列族都由几个文件保存，不同列族的文件是分离的。关系型数据库是基于表格结构和行模式存储的；
* 数据维护：HBase 的更新操作实际上是插入了新的数据，它的旧版本依然会保留，而不是关系型数据库的替换修改；
* 可伸缩性：HBase 这类分布式数据库就是为了这个目的而开发出来的，所以它能够轻松地增加或减少硬件数量，并且对错误的兼容性比较高。而关系型数据库通常需要增加中间层才能实现类似的功能；

**五、案例：搭建Hbase完全分布式数据库系统；**

**案例环境：**

|  |  |  |  |
| --- | --- | --- | --- |
| 系统类型 | IP地址 | 主机名、角色 | 所需软件 |
| Centos 7.4 1708 64bit | 192.168.100.101 | master  hadoop：namenode  hbase：HMaster | hadoop-2.7.6.tar.gz  jdk-8u171-linux-x64.tar.gz  hbase-2.0.1-bin.tar.gz |
| Centos 7.4 1708 64bit | 192.168.100.102 | slave1  hadoop：datanode  hbase：HRegionServer | hadoop-2.7.6.tar.gz  jdk-8u171-linux-x64.tar.gz  hbase-2.0.1-bin.tar.gz |
| Centos 7.4 1708 64bit | 192.168.100.103 | slave2  hadoop：datanode  hbase：HRegionServer | hadoop-2.7.6.tar.gz  jdk-8u171-linux-x64.tar.gz  hbase-2.0.1-bin.tar.gz |

**版本对应：**

****

**下载位置：**http://www.apache.org/index.html#projects-list

**Hbase部署环境：**

**单机模式：**在单台主机运行Hbase；

**伪分布式模式：**HBase只在hadoop的namenode节点运行，与单机模式类似，只是其数据文件可以存储在datanode节点上；

**完全分布式模式：**HBase运行在hadoop集群的多个节点上，通常将HMaster运行在namenode节点上，将HRegionServer运行在datanode节点上；

**案例步骤（保证多个节点之间时间的统一）：**

* 搭建Hadoop分布式存储集群（namenode和datanode）；
* 在master节点安装部署Hbase程序；
* 在master节点配置HBase程序；
* 将master节点的habse程序复制到slave节点；
* 在master节点上开启HBase进程并查看进程；
* 验证slave节点上的进程状态；
* 访问网页，查看HBase运行状态；
* 在master节点登录HBase数据库，查看数据库状态；
* HBase数据库中基本管理操作；
* MapReduce结合HBase查询表中行数；
* **搭建Hadoop分布式存储集群（namenode和datanode）；**
* **在master节点安装部署Hbase程序；**

[root@master ~]# ls hbase-2.0.1-bin.tar.gz

hbase-2.0.1-bin.tar.gz

[root@master ~]# tar zxvf hbase-2.0.1-bin.tar.gz

[root@master ~]# mv hbase-2.0.1 /usr/local/hbase

[root@master ~]# ls /usr/local/hbase

bin conf hbase-webapps lib NOTICE.txt RELEASENOTES.md

CHANGES.md docs LEGAL LICENSE.txt README.txt

[root@master ~]# chown hadoop:hadoop /usr/local/hbase/ -R

* **在master节点配置HBase程序；**

[root@master ~]# su - hadoop

[hadoop@master ~]$ vi /usr/local/hbase/conf/hbase-site.xml ##HBase站点相关配置文件

<configuration>

<property>

<name>hbase.rootdir</name>

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

<description>配置HRegionServer的数据库文件存放目录</description>

</property>

<property>

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

<value>true</value>

<description>配置HBase为完全分布式方式</description>

</property>

<property>

<name>hbase.master</name>

<value>master:60000</value>

<description>配置HMaster的地址和端口</description>

</property>

<property>

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

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

<description>配置zookeeper集群服务器的位置</description>

</property>

</configuration>

[hadoop@master ~]$ vi /usr/local/hbase/conf/hbase-env.sh ##HBase变量配置文件

export JAVA\_HOME=/usr/local/java

export HADOOP\_HOME=/usr/local/hadoop

export HBASE\_HOME=/usr/local/hbase

export HBASE\_MANAGES\_ZK=true

注解：export HBASE\_MANAGES\_ZK=true此配置项意为开启habse内置的zookeeper进程，使其随HBase进程一同启动；

[hadoop@master ~]$ vi /usr/local/hbase/conf/regionservers ##HBase的节点

slave1

slave2

* **将master节点的habse程序复制到slave节点；**

[root@slave1 ~]# mkdir /usr/local/hbase

[root@slave1 ~]# chown hadoop:hadoop /usr/local/hbase/

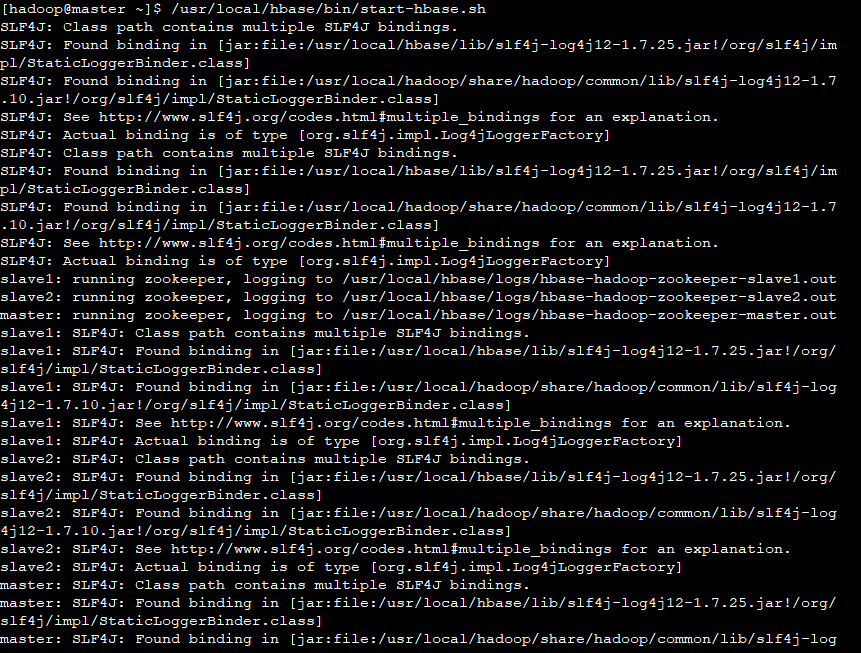
[root@slave2 ~]# mkdir /usr/local/hbase

[root@slave2 ~]# chown hadoop:hadoop /usr/local/hbase/

[hadoop@master ~]$ scp -r /usr/local/hbase/\* hadoop@slave1:/usr/local/hbase

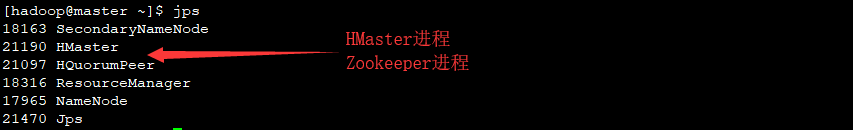
[hadoop@master ~]$ scp -r /usr/local/hbase/\* hadoop@slave2:/usr/local/hbase

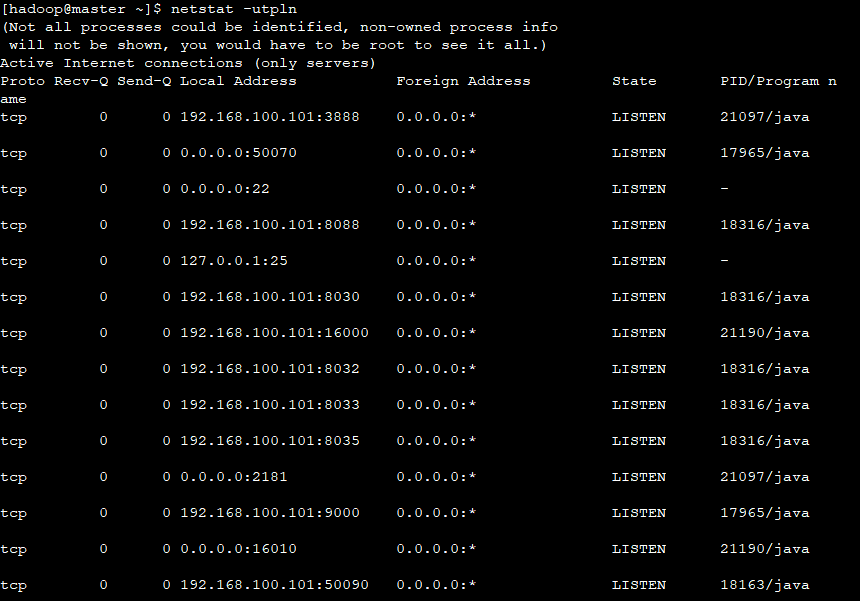
* **在master节点上开启HBase进程并查看进程；**



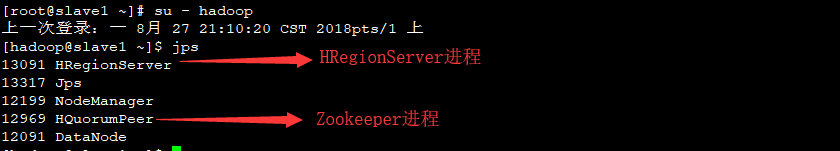
注解：如若启动hbase时，出现：错误：找不到或无法加载主类；

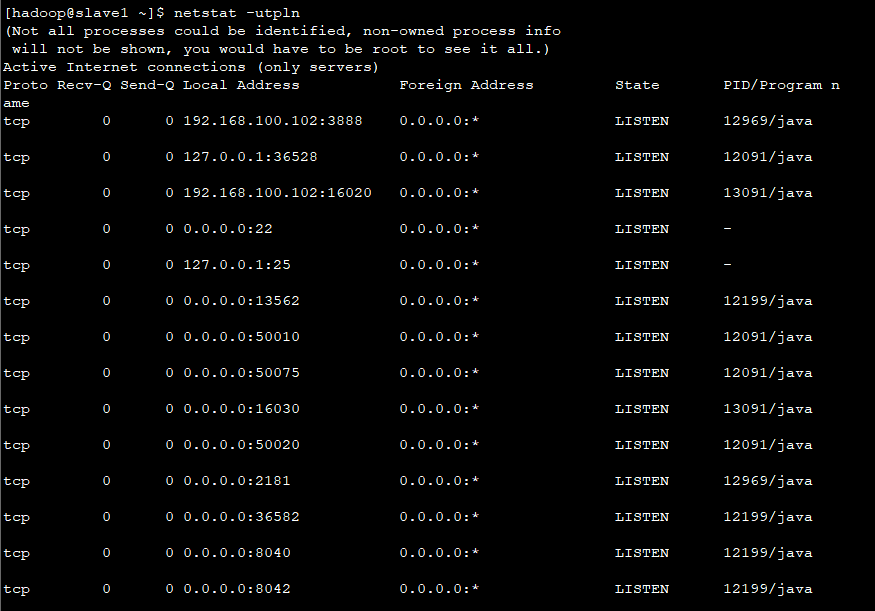
由于habse版本与hadoop版本导致，或者环境变量导致；

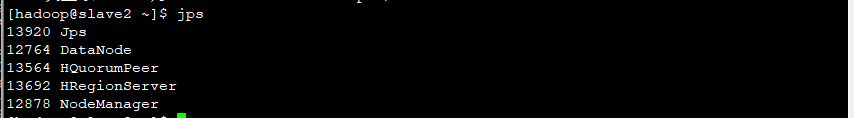


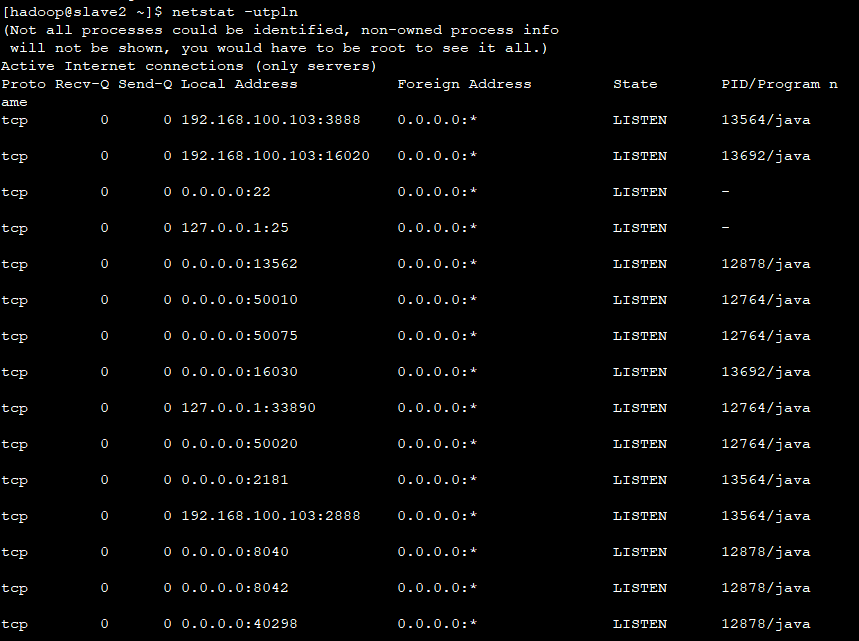


* **验证slave节点上的进程状态；**



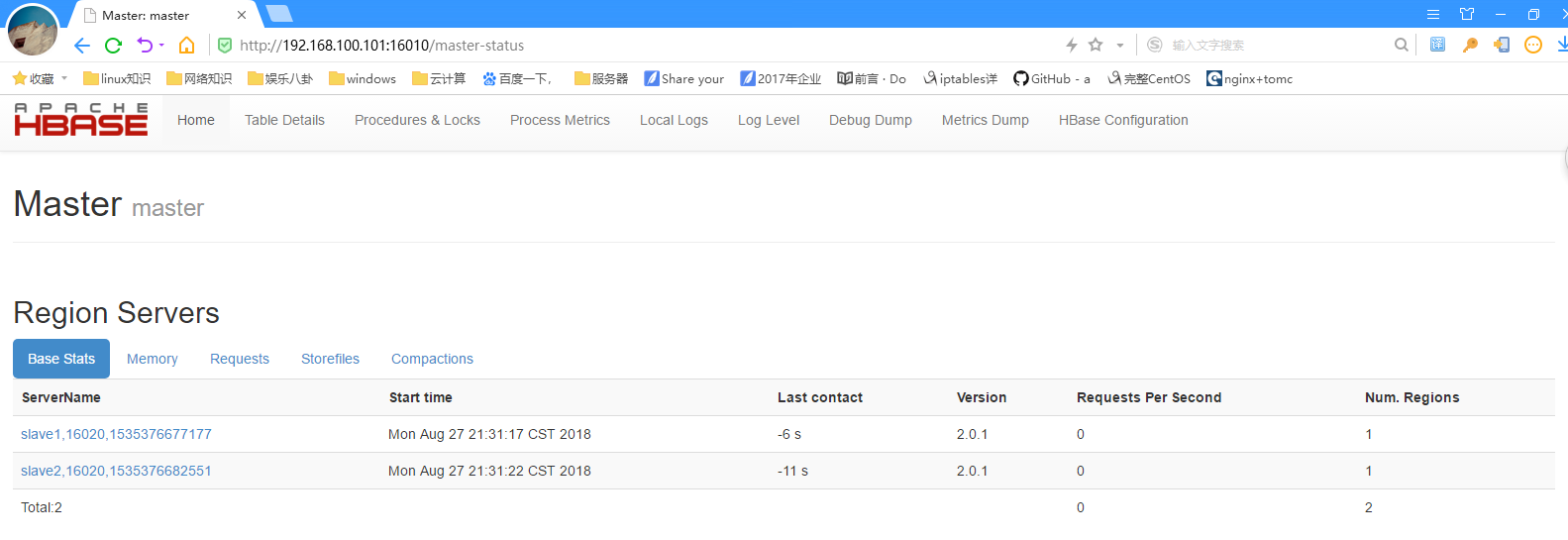


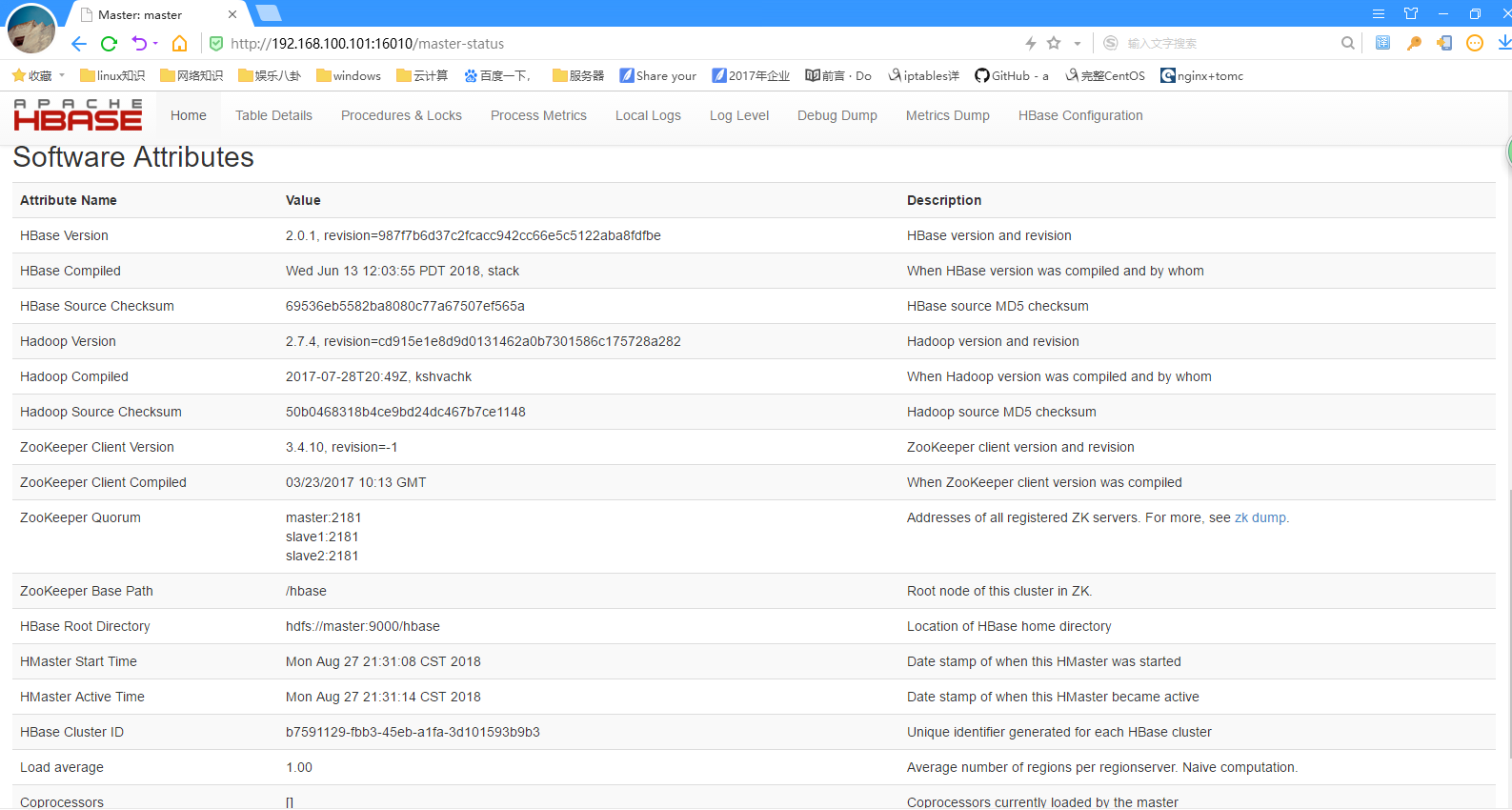




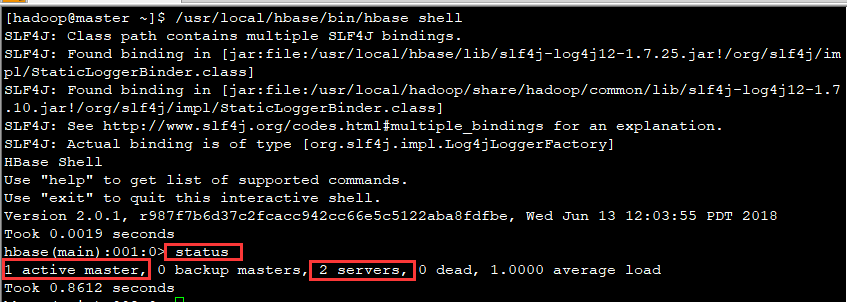
* **访问网页，查看HBase运行状态；**

http://192.168.100.101:16010

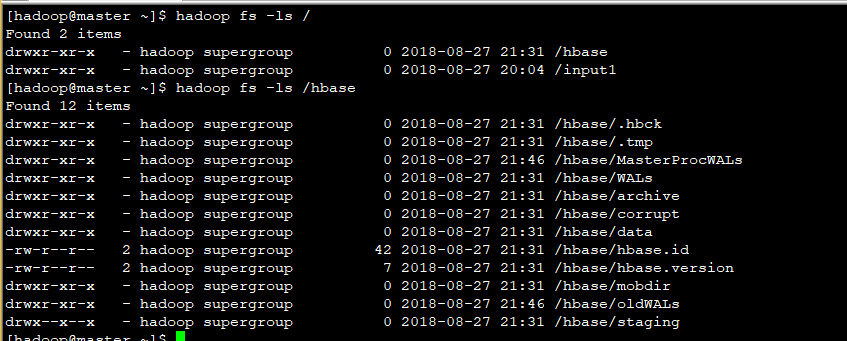




* **在master节点登录HBase数据库，查看数据库状态；**



* **在master节点访问hadoop存储中数据，验证数据文件状态；**



* **HBase数据库中基本管理操作；**

[hadoop@master ~]# /usr/local/hbase/bin/hbase shell

hbase(main):001:0> status ##查看状态

1 active master, 0 backup masters, 2 servers, 0 dead, 1.0000 average load

Took 0.8818 seconds

hbase(main):002:0> create 'class','age','chengji' ##创建表，语法：create 表名 列族 列键

Created table class

Took 1.5186 seconds

=> Hbase::Table - class

hbase(main):003:0> list ##查看所有表

TABLE

class

1 row(s)

Took 0.0940 seconds

=> ["class"]

hbase(main):004:0> describe 'class' ##查看表的详细信息

Table class is ENABLED

class

COLUMN FAMILIES DESCRIPTION

{NAME => 'age', VERSIONS => '1', EVICT\_BLOCKS\_ON\_CLOSE => 'false', NEW\_VERSION\_BEHAVIOR => 'f

alse', KEEP\_DELETED\_CELLS => 'FALSE', CACHE\_DATA\_ON\_WRITE => 'false', DATA\_BLOCK\_ENCODING =>

'NONE', TTL => 'FOREVER', MIN\_VERSIONS => '0', REPLICATION\_SCOPE => '0', BLOOMFILTER => 'ROW'

, CACHE\_INDEX\_ON\_WRITE => 'false', IN\_MEMORY => 'false', CACHE\_BLOOMS\_ON\_WRITE => 'false', PR

EFETCH\_BLOCKS\_ON\_OPEN => 'false', COMPRESSION => 'NONE', BLOCKCACHE => 'true', BLOCKSIZE => '

65536'}

{NAME => 'chengji', VERSIONS => '1', EVICT\_BLOCKS\_ON\_CLOSE => 'false', NEW\_VERSION\_BEHAVIOR =

> 'false', KEEP\_DELETED\_CELLS => 'FALSE', CACHE\_DATA\_ON\_WRITE => 'false', DATA\_BLOCK\_ENCODING

=> 'NONE', TTL => 'FOREVER', MIN\_VERSIONS => '0', REPLICATION\_SCOPE => '0', BLOOMFILTER => '

ROW', CACHE\_INDEX\_ON\_WRITE => 'false', IN\_MEMORY => 'false', CACHE\_BLOOMS\_ON\_WRITE => 'false'

, PREFETCH\_BLOCKS\_ON\_OPEN => 'false', COMPRESSION => 'NONE', BLOCKCACHE => 'true', BLOCKSIZE

=> '65536'}

2 row(s)

Took 0.1701 seconds

hbase(main):012:0> put 'class','tom','age','18' ##添加数据，语法：put 表名 行键 列键 值

Took 0.1784 seconds

hbase(main):013:0> put 'class','marry','age','20'

Took 0.0262 seconds

hbase(main):014:0> scan 'class' ##扫描class表中数据

ROW COLUMN+CELL

marry column=age:, timestamp=1535528846020, value=20

tom column=age:, timestamp=1535528825217, value=18

2 row(s)

Took 0.0628 seconds

hbase(main):017:0> put 'class','tom','chengji:math','95' ##插入数据

Took 0.0217 seconds

hbase(main):018:0> put 'class','tom','chengji:english','90'

Took 0.0100 seconds

hbase(main):019:0> put 'class','marry','chengji:math','85'

Took 0.0130 seconds

hbase(main):020:0> put 'class','marry','chengji:english','90'

Took 0.0085 seconds

hbase(main):021:0> scan 'class'

ROW COLUMN+CELL

marry column=age:, timestamp=1535528846020, value=20

marry column=chengji:english, timestamp=1535529132585, value=90

marry column=chengji:math, timestamp=1535529119078, value=85

tom column=age:, timestamp=1535528825217, value=18

tom column=chengji:english, timestamp=1535529101465, value=90

tom column=chengji:math, timestamp=1535529089638, value=95

2 row(s)

Took 0.0120 seconds

hbase(main):033:0> scan 'class',{COLUMN=>'chengji:math',LIMIT=>1} ##根据条件查找，显示一行

ROW COLUMN+CELL

marry column=age:, timestamp=1535528846020, value=20

marry column=chengji:english, timestamp=1535529132585, value=90

marry column=chengji:math, timestamp=1535529119078, value=85

1 row(s)

Took 0.0456 seconds

hbase(main):038:0> get 'class','tom' ##获取表中数据，语法：get 表名 行键

COLUMN CELL

age: timestamp=1535528825217, value=18

chengji:english timestamp=1535529101465, value=90

chengji:math timestamp=1535529089638, value=95

1 row(s)

Took 0.0125 seconds

hbase(main):042:0> get 'class','tom',{COLUMN=>'age:'} ##根据条件获取表中数据，语法：get 表名 行键 {COLUMN=>列族}

COLUMN CELL

age: timestamp=1535528825217, value=18

1 row(s)

Took 0.0188 seconds

hbase(main):043:0> get 'class','tom','age:' ##根据条件获取表中数据，同上

COLUMN CELL

age: timestamp=1535528825217, value=18

1 row(s)

Took 0.0171 seconds

hbase(main):044:0> get 'class','tom','chengji:english'

COLUMN CELL

chengji:english timestamp=1535529101465, value=90

1 row(s)

Took 0.0162 seconds

hbase(main):045:0> delete 'class','tom','chengji:english' ##删除表中数据记录，语法：delete 表名 行键 列键

Took 0.0367 seconds

hbase(main):046:0> get 'class','tom','chengji:english' ##获取表中数据记录，无法获取

COLUMN CELL

0 row(s)

Took 0.0226 seconds

hbase(main):047:0> get 'class','tom' ##获取表中tom此行键的所有内容

COLUMN CELL

age: timestamp=1535528825217, value=18

chengji:math timestamp=1535529089638, value=95

1 row(s)

Took 0.0106 seconds

hbase(main):048:0> disable 'class' ##删除表之前，需要先将表关闭disable

Took 0.8495 seconds

hbase(main):049:0> drop 'class' ##删除表

Took 0.4907 seconds

hbase(main):050:0> list ##查看所有表

TABLE

0 row(s)

Took 0.0086 seconds

=> []

hbase(main):051:0> exit

* **MapReduce结合HBase查询表中行数；**

[hadoop@master ~]$ cp /usr/local/hbase/conf/hbase-site.xml /usr/local/hadoop/etc/hadoop/

[hadoop@master ~]$ vi /usr/local/hadoop/etc/hadoop/hadoop-env.sh

export HADOOP\_CLASSPATH=$HADOOP\_CLASSPATH:/usr/local/hbase/lib/\*

[hadoop@master ~]$ scp -r /usr/local/hadoop/etc/hadoop/hadoop-env.sh hadoop@slave1:/usr/local/hadoop/etc/hadoop/

[hadoop@master ~]$ scp -r /usr/local/hbase/conf/hbase-site.xml hadoop@slave1:/usr/local/hbase/conf/

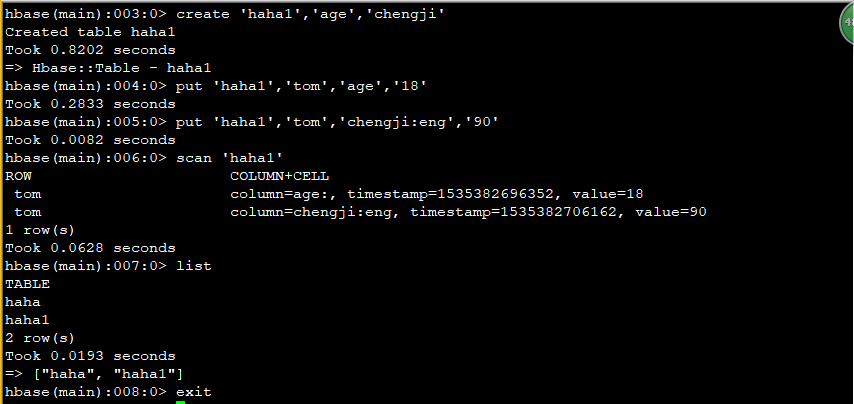
[hadoop@master ~]$ scp -r /usr/local/hadoop/etc/hadoop/hadoop-env.sh hadoop@slave2:/usr/local/hadoop/etc/hadoop/

[hadoop@master ~]$ scp -r /usr/local/hbase/conf/hbase-site.xml hadoop@slave2:/usr/local/hbase/conf/

[hadoop@master ~]$ hadoop jar /usr/local/hbase/lib/hbase-server-2.0.1.jar

RunJar jarFile [mainClass] args...

[hadoop@master ~]$ /usr/local/hbase/bin/hbase shell



[hadoop@master ~]$ /usr/local/hbase/bin/hbase org.apache.hadoop.hbase.mapreduce.RowCounter 'haha1'

