HBASE总结：

这三个hbase节点，应该安装在有启动zookeeper的三个几点上

安装总结：

1.解压后，进入conf配置 hbase-env.sh

三处：

1.java\_home

2.export HBASE\_CLASSPATH=/home/tools/hadoop-2.5.1/etc/hadoop 路径是hadoop的配置文件路径

3.export HBASE\_MANAGES\_ZK=false 是否使用内置zookeeper



2.把hdfs配置文件hdfs.site.xml拷贝到conf下

3.配置hbase-site.xml

<property>

<name>hbase.rootdir</name>

<value>hdfs://sxt/hbase</value> 这个sxt是hadoop的core-site.xml中的hdfs名称

</property>

<property>

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

<value>true</value>

</property>

<property>

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

<value>node01,node02,node03</value>

</property>

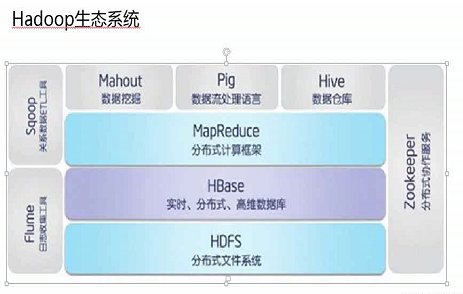
4.修改regionservers 配置三个节点名

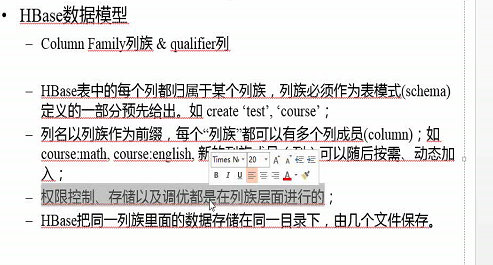
5.拷贝hbase文件到另外两个节点下。



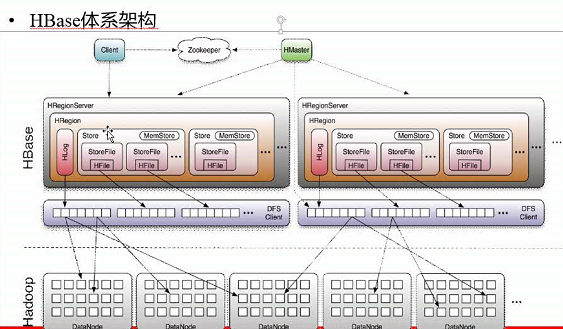
解除hdfs的安全模式

1. bin/hbasestart.sh 启动后可以看三个节点的jps，hmaster和hserver都有了。端口号60010。
2. bin/hbase shell 进行数据库操作。





在存储的时候，一个列族就是一个目录。



Cell单元格（列里面的内容）内容是二进制字节码存储的。（编码格式由客户自己设置。）

hMaster不存元数据，它指挥zookeeper存元数据，并存储所有region的寻址入口。

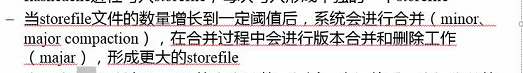
Client请求hmaster。它同意后，client和regin server连接。

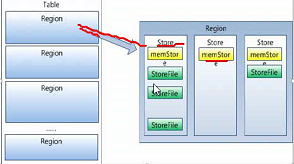
一个reginserver下有多个regin，维护、处理region与客户之间的IO请求。负责切分变的过大的region。可以看做是hbase开启的服务。

一个regin是一个表（表小）或者一个大表的一部分。Regin存一个表的数据，数据满了，就多分一个regin存这个表。Regin个数不固定（根据表的大小。）

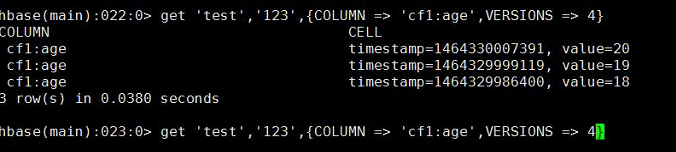
一个regin内的storefile固定，一个storeflie代表一个列族。（在hdfs里，点开regin名，就是列族名，点开列族名，就是storefile名）可以看一下。

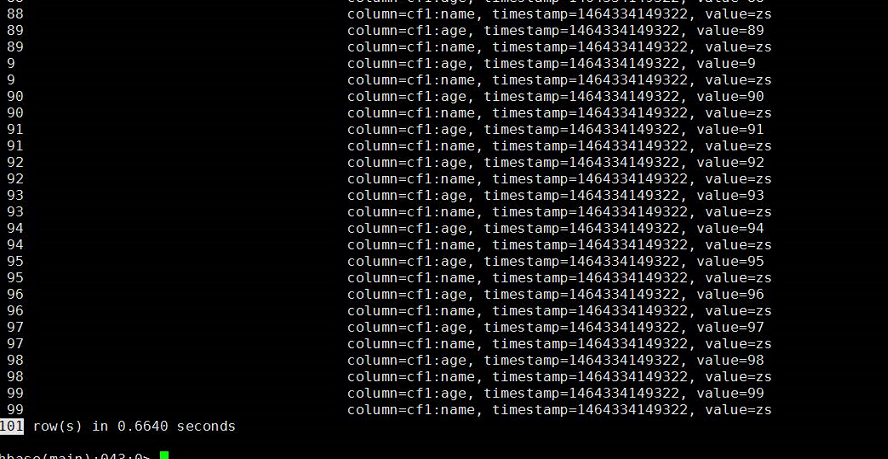
一个store里有一个mem。满了就往hdfs里存。存成一个hfile（storefile和hfile是一个东西）。mem新来的往新的hfile里存。



region内部结构。

操作：





操作命令行：

建表：creat ‘test’，‘cf1’，‘cf2’ 表名，两个列族名

查看表：desc ‘test’

插入数据：put ‘test’，‘123’，‘cf1：name’‘zhangsan’

表名 rowkey 列族，列族中的属性 属性值

查一个数据全部：get‘表名’，‘rowkey’

查一个数据的精确内容：get ‘表名’，‘123’，‘cf1：name’查rowkey为123的数据的name属性

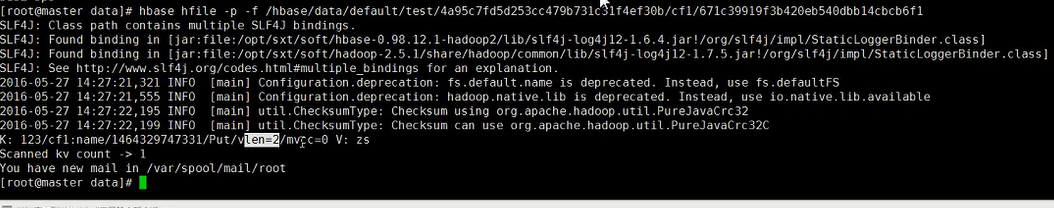
查看表的全部：scan ‘表名’

修改表属性 alter ‘test’，{NAME=>’cf1’，VERIONS=>5}把cf1列族的版本改为5

Get ‘test’，‘rowkkey’，{COLUMN=>’cf1:age’,VERSIONS=>4} 就能查看这个属性，最近的4个版本了。

Flush ‘表名（regin名）’ 就能让mem中的内容，进入storefile。（mem装满或者过一段时间，就自动进入了，也可以这样手动刷）

Hbase -p -f /hdfs中文件的路径。就能看到数据全体了。



命令行平时用的少，用的时候：

Help可以查看命令。

在输入命令，就可以查看具体命令使用方式。



一：建表部门表：key是部门编号 列族：一个。

列族中的列：parent：部门编号。

1.查询parent为null的部门。

2.查询parent为那个部门的所有部门。

3.修改指定key的parent属性。

二：建表1学生表：key是学生学号。 列族：一个。建表2课程表：key是课程。列族：一个。

表1中的列：课程名集合。 表二中的列：学生学号集合。

1. 查询表1对应学生key的value。
2. 查询表2对应课程key的value。
3. 修改表1，2对应key的value。

三：2个表。1：key为用户，列有2个，关注集合，粉丝集合。

2：key为微博内容，以时间为开头，列有1个，用户名。

题1代码：

建表：

public static TableName table = TableName.valueOf("dept");

public void createTable() throws IOException {

HBaseAdmin admin = new HBaseAdmin(*conf*);  
 if(admin.tableExists(*table*)){  
 admin.disableTable(*table*);  
 admin.deleteTable(*table*);  
 }  
 HTableDescriptor hd = new HTableDescriptor(*table*);  
 HColumnDescriptor hc = new HColumnDescriptor("cf1");  
 hc.setBlockCacheEnabled(true);  
 hc.setBlocksize(12800000);  
 hc.setMaxVersions(10);  
 hd.addFamily(hc);  
 admin.createTable(hd);  
}

往里面添加部门（目前有三层部门，每层部门数随机）：

public void insert() throws IOException {  
 HTable hTable = new HTable(*conf*, *table*);  
 List<Put> list = new ArrayList<Put>();  
 int i= (int) ( 50\* Math.*random*()); //一级部门  
 for (int a = 0; a <i ; a++) {  
 Put put = new Put(("1"+"\_"+String.*valueOf*(a)).getBytes());  
 list.add(put);  
 }  
  
 int j= (int) ( 50\* Math.*random*());//二级部门  
 for (int b= 0; b < j; b++) {  
 int n = (int) ( i\* Math.*random*());  
 Put put = new Put(("2"+"\_"+String.*valueOf*(b)).getBytes());  
 put.add("cf1".getBytes(), "parent".getBytes(), String.*valueOf*(n).getBytes());  
 list.add(put);  
 }  
  
 int k= (int) ( 50\* Math.*random*());//三级部门  
 for (int c = 0; c < k; c++) {  
 int n = (int) ( j\* Math.*random*());  
 Put put = new Put(("3"+"\_"+String.valueOf(c)).getBytes());  
 put.add("cf1".getBytes(), "parent".getBytes(), String.*valueOf*(n).getBytes());  
 list.add(put);  
 }  
  
 hTable.put(list);

查询parent为null的部门：

@Test  
public void scanByCloumn() throws IOException {  
 HTable hTable = new HTable(*conf*, *table*);  
 Scan scan = new Scan();  
 FilterList list = new FilterList(FilterList.Operator.*MUST\_PASS\_ALL*);  
 SingleColumnValueFilter sf = new SingleColumnValueFilter("cf1".getBytes(),"parent".getBytes(), CompareFilter.CompareOp.*EQUAL*,"".getBytes());  
 list.addFilter(sf);  
 scan.setFilter(list);  
 ResultScanner results = hTable.getScanner(scan);  
 Iterator<Result> it = results.iterator();  
 while(it.hasNext()){  
 Result result = it.next();  
 Cell cell = result.getColumnLatestCell("cf1".getBytes(), "parent".getBytes());  
 System.*out*.println(new String(CellUtil.*cloneRow*(cell),"UTF-8"));  
 System.*out*.println(new String(CellUtil.*cloneValue*(cell),"UTF-8"));  
 }  
}

经老师讲解,表设计应该以查询为准。上面的设计思想，依旧和关系型数据库的设计相同！！

1. 部门表

Rowkey cf1 cf2

部门编号 基本信息 列属性为他的子部门，value：1（标示值无意义）

这样设计，修改时，定位rowkey，修改一个单元格即可，速度快。

1. 学生表

Rowkey cf1 cf2

Sid 基本信息 列属性名设置为课程，value：1

课程表

Rowkey cf1 cf2

Cid 基 列属性设置为学生，value：1

这样，修改一名学生的课程时，只需要修改两个单元格就行，速度很快。