Hbase学习

命令行：

start-hbase.sh

create 'testtable','colfam1'

list

list 'testtable'

put 'testtable','myrow-1','colfam1:q1','value-1'

scan 'testtable'

delete 'testtable','myrow-1','colfam1:q1'

disable 'testtable'

drop 'testtable'

stop-hbase.sh

小知识：

hbase目前只能依赖特定版本hadoop，主要原因之一是hbase和hadoop的远程调用过程Remote Procedure Call是版本协议化的

需要注意的hbase运行系统配置:

1、需要同步时间

2、增大文件描述符限制

用户API：

基本的行增加工作：

Configuration conf = HBaseConfiguration.create();

HTable table = new HTable (conf,”testtable”);

Put put = new put(Bytes.toBytes(“row1”));

put.add(Bytes.toBytes(“colfam1”),Bytes.toBytes(“qual1”),Bytes.toBytes(“val1”));

缓冲区的概念：

1、用户的缓冲机制：put实例在一开始的时候被保留在内存中，当发起commit命令的时候才通过一次RPC批量的发送到服务器

2、设置写缓冲机制：

table.setAutoFlush(false);

3、强制把数据写到服务端：

void flushCommits() throws IOException

4、配置缓冲区大小

long getWriteBufferSize();

void setWriteBufferSize(long writeBufferSize) throws IOException

5、为了避免为每个HTable实例都设定缓冲区的大小，在配置文件中添加一个较大的预设值：

<property>

<name>hbase.client.write.buffer</name>

<value>20971520</value>

</property>

这是把缓冲区设置为20M的配置

6、在强制调用刷缓冲区方法、缓冲区满、设置缓冲区大小且目前缓冲区出限制以及HTable类被close的时候都会刷写缓冲区。

列表的使用：

使用put列表可以批量的刷写数据，使用方法就是先定义一个列表，然后在调用列表的add方法向其中增加元素，在直接将列表作为put的参数：

List<Put> puts = new ArrayList<Put>();

Put put1 = new Put...

puts.add(put1);

Put put2 = …

puts.add(put2);

table.put(puts);

原子操作：

table.checkAndPut可以保证写操作的原子性；

Put put1 = …

put1.add...

boolean res1 = table.checkAndPut(Bytes.toBytes(“row1”),

Bytes.toBytes(“colfam1”),Bytes.toBytes(“qual1”),null,put1);

检查列族，列，行是否存在，且值为空，如果都满足，则put

get的用法：

利用get从HBase 中获取数据：

Configuration conf = HbaseConfiguration.create();

HTable table = new HTable (conf, “testtable”);

Get get = new Get(Bytes.toBytes(“row1”));

get.addColumn(Bytes.toBytes(“colfam1”),Bytes.toBytes(“qual1”));

Result result = table.get(get);

byte[] val = result.getValue(Bytes.toBytes(“colfam1”),

Bytes.toBytes(“qual1”));

System.out.println(“Value: “ + Bytes.toString(val));

result类的用法：

用户使用get()方法获取的数据存储在一个result实例中

getValue、value、getRow...等等方法以各种方式返回了相应的数据

delete的用法，和get、put等相类似

batch的使用，batch的目的就是批量处理，可以把put、get、delete等操作合在一个batch上。

lock的使用

lock用来对行进行锁定

scan方法

可以以规定行等方式扫描表

扫描器缓存：为了一次RPC获取更多行的数据而存在；

有两种存在的方式，一种是存在于表，那么这个表的所有扫描器实例都会生效，

另一种是存在与扫描器，那么这个扫描器实例会被影响。

缓存：缓存决定了行数，每次传送缓存个数的行数；

批量：批量决定传送的列数；

所以RPC次数 = 行数\*列数/min（批量，行数）/缓存再加一；

在HBase中，所谓的一列就是指一行中的一个单元格，而非所有行的那一列，因为每个单元格还需要保存很多个版本，所以一列仍然有很多数据。

高级API:

过滤器：

有多种过滤器，都继承了Filter借口或FilterBase抽象类

和过滤器一起使用的还有比较器：CompareFilter，比较只能正面指定需要过滤的值，而不是取反。

匹配相关的比较器由于只返回真假，所以只能设置为EQUE与否，而不能设置大小比较。

以行过滤器为例；

Filter filter1 = new RowFilter(CompareFilter.CompareOp.LESS\_OR\_EQUAL,

new BinaryComparator(Bytes.toBytes(“row-22”)));

scan.setFilter(filter1);

Resultscanner scanner1 = table.getScanner(scan);

for (Result res : scanner1) {

System.out.println(res);

}

scanner1.close();

问题点：既然不可用不存在的列族，那么如何创建新的列族？

答：只能在创建table的时候规定列族

炉石传说：

1、一费的研究：

1、如果是先手

1、如果开始的时候没有合适的低费，是留不厉害的低费，还是留厉害的高费？

2、

2、如果是后手

职业选择：

需要针对防战士做出反应

崩盘点：

1、5费上个3/6嘲讽被冲死