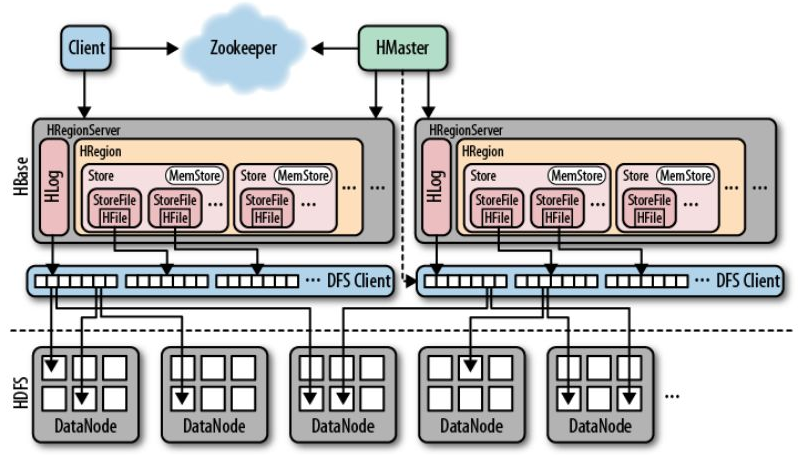
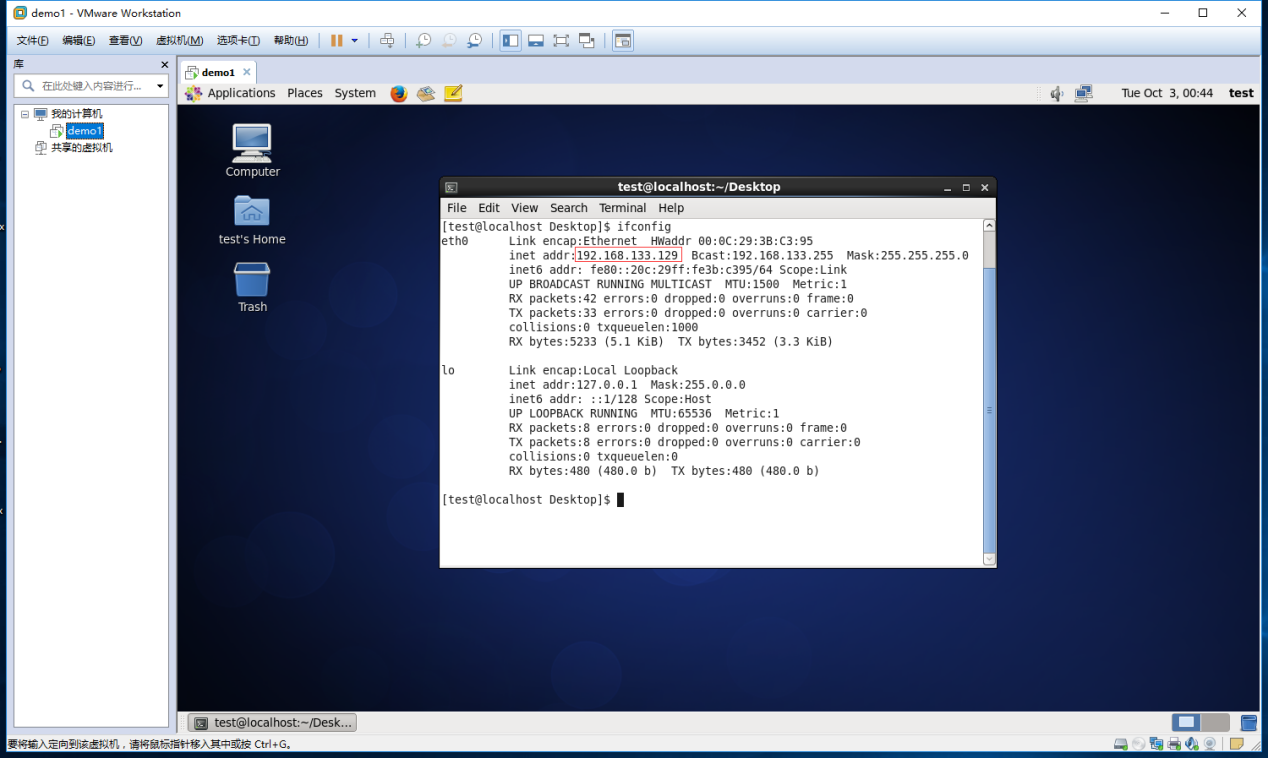
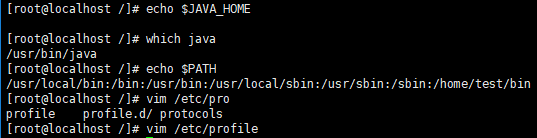
HBase说明文档：<http://hbase.apache.org/book.html>



1. 基于虚拟机安装



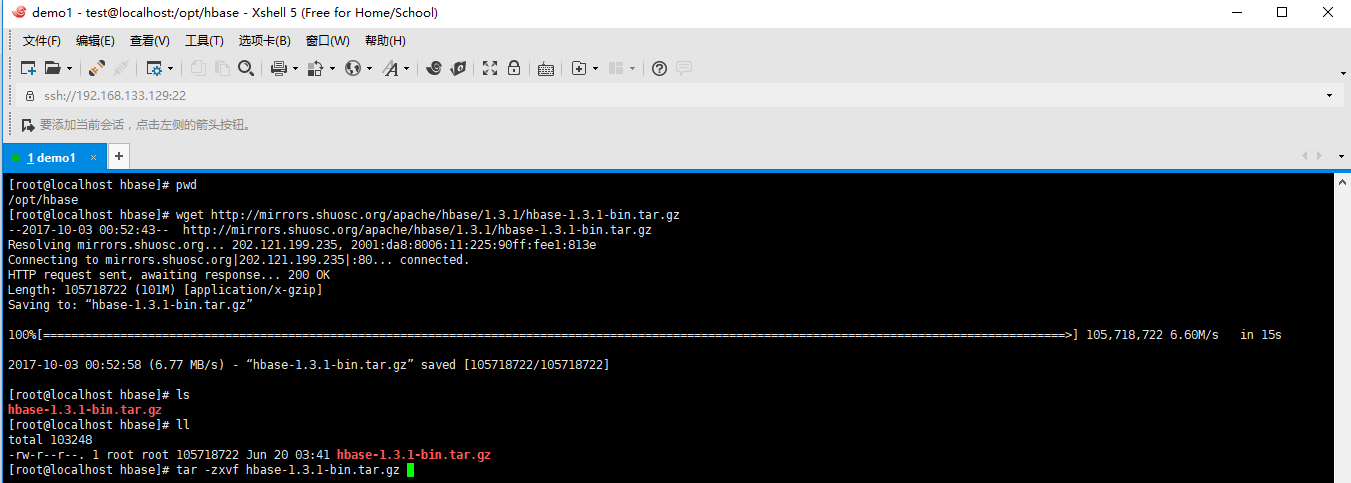
1. 安装JAVA



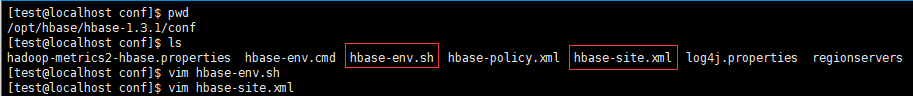


键入source /etc/profile让配置生效

2、安装HBase



主要配置两个文件



二、非集群，快速开始配置

1、hbase-site.xml



分布式的提供简单参考，具体见官方说明文档：

<property>  
 <name>hbase.rootdir</name>  
 <value>hdfs://192.168.1.11:8020/hbase</value>  
</property>

<property>  
 <name>hbase.tmp.dir</name>  
 <value>/usr/local/bigdata/hbase-0.98.6-cdh5.3.6/data/tmp</value>  
</property>  
<property>  
 <name>hbase.zookeeper.property.dataDir</name>  
 <value>/usr/local/bigdata/hbase-0.98.6-cdh5.3.6/data/zk</value>  
</property>

<property>  
 <name>hbase.cluster.distributed</name>  
 <value>true</value>  
</property>

<property>

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

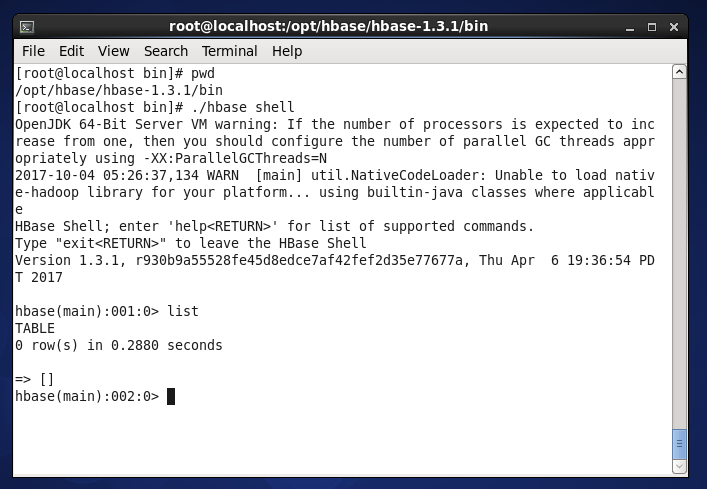
<value>192.168.1.11</value>

</property>

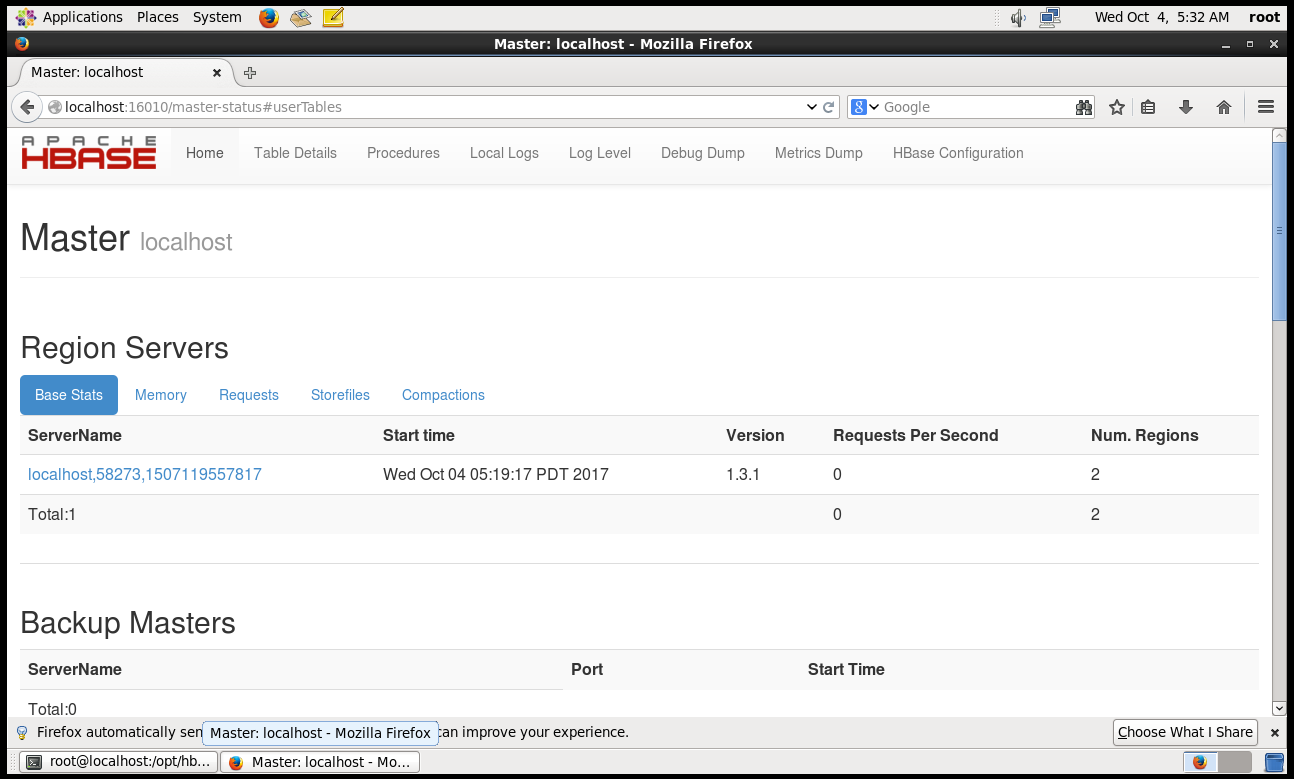
2、hbase-env.sh



简单看下是否配置成功：



页面查看：http://localhost:16010

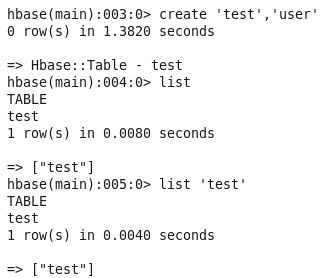


创建命名空间：

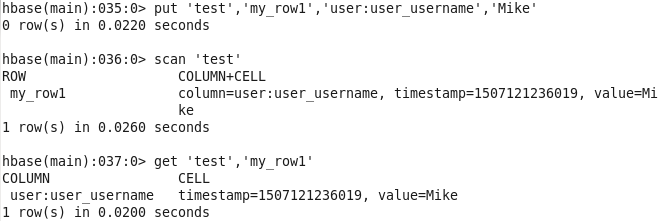
create\_namespace 'lrk'

可以根据创建好的命名空间建表：create 'lrk:test','user'

建表：



新增及查询数据：



1. JavaAPI简单体验

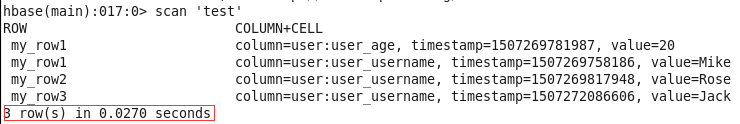
1、pom.xml配置



2、Java代码编写



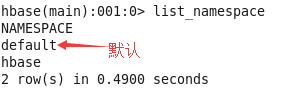
看下查询结果：



注：①查出来是有四条记录的，但是其实是三行数据，如何合并四条记录为三条记录呢？如果是根据rowKey查询，遇到这种存储结果确实不大好处理，目前是代码判断处理的。②这里我是直接通过mvn clean install打成jar包然后放到虚拟机执行java -jar hbase.jar的

1. 补充
2. 命名空间

默认都在default里



1. 表设计核心

尽量围绕Row Key来设计及前缀匹配规则，Row Key可以单个搜索、可以范围搜索、可以全表搜索

1. 其它语言（如C++/php等）接入HBase

可以采用Rest或Thrift，具体参见官方说明文档

1. 建立多个列簇

在建表时：create 'test','a1','a2'，表示表test建立了两个列簇a1和a2，添加数据的话：

put 'test2','row1','a1:key1','value1'

建表后（不推荐）：

disable '表名'  
alter '表名', NAME => '列名', VERSIONS =>3  
enable '表名'

1. 删除表

disable 'test'

drop 'test'

1. 清空表

truncate ‘表名’

7、事物处理

建议采用batch操作

1. 分页

常规分页可能需要包含总条数（总页数），但是如果HBase分页也要获取总条数（总页数）的话，除了条件查询的开销外还需要扫描总条数的开销，HBase并未提供类似mysql的count查询，在数据量非常大的情况下会极其耗时，因此用HBase进行分页，建议不显示总条数（总页数）。下面阐述两种做法的关键代码：

A、包含总条数（总页数）的做法

Scan scan = new Scan();

if(filters.size()!=0){

FilterList filterList = new FilterList(FilterList.Operator.MUST\_PASS\_ALL,filters);

scan.setFilter(filterList);

}

ResultScanner resultScanner = table.getScanner(scan);

long count = 0;

long currentStart =

(comprehensiveQueryBean.getCurrentPage()-1)\*comprehensiveQueryBean.getPageSize();

long currentEnd = currentStart+comprehensiveQueryBean.getPageSize();

QueryResultBean queryResultBean = new QueryResultBean();

Page page = new Page();

for(Result result:resultScanner){

if(currentStart<=count&&count<currentEnd){//表示分页

//低效获取当前页数需要的数据

}

count++;//表示统计总条数

}

page.setPageSize(comprehensiveQueryBean.getPageSize());

page.setCurrentPage(comprehensiveQueryBean.getCurrentPage());

page.setTotalSize(count);

if(page.getTotalSize()%comprehensiveQueryBean.getPageSize()==0){ page.setTotalPageSize(page.getTotalSize()/comprehensiveQueryBean.getPageSize());

}else{ page.setTotalPageSize(page.getTotalSize()/comprehensiveQueryBean.getPageSize()+1);

}

queryResultBean.setList(resultList);

queryResultBean.setPage(page);

return queryResultBean;

可以看到所有的处理都在后端进行的

B、不含总条数（总页数）的做法

//每次都预查100条数据

PageFilter pageFilter = new PageFilter(comprehensiveQueryBean.getPageSize()\*10);

filters.add(pageFilter);

Scan scan = new Scan();

FilterList filterList = new FilterList(FilterList.Operator.MUST\_PASS\_ALL,filters);

scan.setFilter(filterList);

String startRow = comprehensiveQueryBean.getStartRow();

ResultScanner resultScanner = null;

if(startRow==null||comprehensiveQueryBean.getCurrentPage()==1){//表示第一页

resultScanner = table.getScanner(scan);

}else{//表示非第一页

scan.setStartRow(Bytes.toBytes(startRow));

resultScanner = table.getScanner(scan);

}

long currentStart =

(comprehensiveQueryBean.getCurrentPage()-1)\*comprehensiveQueryBean.getPageSize();

QueryResultBean queryResultBean = new QueryResultBean();

Page page = new Page();

for(Result result:resultScanner){

//封装数据

}

page.setPageSize(comprehensiveQueryBean.getPageSize());

page.setCurrentPage(comprehensiveQueryBean.getCurrentPage());

queryResultBean.setList(resultList);

queryResultBean.setPage(page);

return queryResultBean;

后端代码较简单，主要就是一个setStartRow方法，主要复杂点在前端，看下主要代码：

①首先缓存两个变量

var startRow = null;

var cacheMap = {};//缓存每一页的第一条数据的rowKey

②每次重新更换条件搜索或再次以相同条件搜索时，重置两个变量

var startRow = null;

var cacheMap = {};

③处理获得的数据并分页

//进行分页处理

var totalDataCount = getData.list.length;//获取后端一共返回了多少条数据

var shouldShowTotalPage = 1;

if(totalDataCount<10){//总条数<10就是1页

shouldShowTotalPage = 1;

}else if(totalDataCount%10==0){//总条数是10的倍数就是条数除以10

shouldShowTotalPage = parseInt(totalDataCount/10);

}else{//其它就是条数除以10加1

shouldShowTotalPage = parseInt(totalDataCount/10) + 1;

}

var pageContext = [];

currentPage = parseInt(currentPage);

if(currentPage>=1&&currentPage<=5){

//前部处理

for(var i=1;i<=(currentPage-1);i++){

pageContext.push(i);

}

//后部处理

var restShouldShow = 10-currentPage;//剩余应显示数

if(restShouldShow>=shouldShowTotalPage){//剩余应显示数>=总页数,则以总页数为基准

for(var i=currentPage;i<(currentPage+shouldShowTotalPage);i++){

pageContext.push(i);

}else{//剩余应显示数<总页数,则以剩余应显示数为基准

for(var i=currentPage;i<=(currentPage+restShouldShow);i++){

pageContext.push(i);

}

}

}else{

//前部处理(永远只显示4个)

for(var i=(currentPage-4);i<currentPage;i++){

pageContext.push(i);

}

//后部处理

var restShouldShow = 6;//10-4(这里应该永远是6);//剩余应显示数

//剩余应显示数>=总页数,则以总页数为基准

if(restShouldShow>=shouldShowTotalPage){

for(var i=currentPage;i<(currentPage+shouldShowTotalPage);i++){

pageContext.push(i);

}

}else{//剩余应显示数<总页数,则以剩余应显示数为基准

for(var i=currentPage;i<(currentPage+restShouldShow);i++){

pageContext.push(i);

}

}

}

//显示页数

var pageFy = '';

for(var i=0;i<pageContext.length;i++){

var pageNum = pageContext[i];

if(currentPage==pageNum){

pageFy += "<li class='pg cur' onclick='chosePage(this)'>"+pageNum+"</li>";

}else{

pageFy += "<li class='pg' onclick='chosePage(this)'>"+pageNum+"</li>";

}

}

$(".prev").after(pageFy);

//上一页和下一页显示逻辑

if(currentPage==1){

$("#beforePage").hide();

}else{

$("#beforePage").show();

}

if(totalDataCount>10){

$("#afterPage").show();

}else{

$("#afterPage").hide();

}

//保存从当前页开始所有可见范围内的startRowKey

//这一步随着页数的增加可能缓存数据很多，如果要优化可以比对pageContext 的值来剔除

var cut = 0;

for(var i=0;i<totalDataCount;i=i++){

cacheMap[currentPage+cut] = getData.list[i].rowKey;

cut++;

i+=10;

}

④点击上一页、下一页和某页的处理

if(cacheMap[currentPage]!=null&&cacheMap[currentPage]!=undefined){

startRow = cacheMap[currentPage];

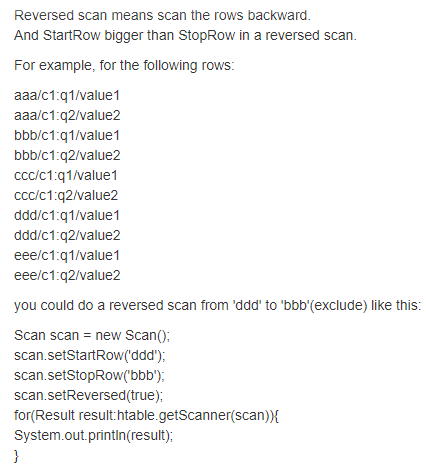
}else{

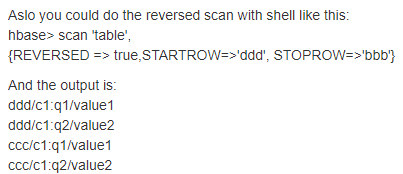
startRow = null;

}

1. 倒序

直接提供了scan.setReversed(true)方法，注意scan.setStartRow()也应该是倒序的，详见如下：





如果又有分页、又有过滤条件，此时用倒序最大的问题就是需要知道stratRow，即最后一条数据，如果不设置startRow，直接使用倒序，查询效率是非常低的，因此使用倒序需谨慎