# Python操作HBase、Java

本文主要是描述Coprocessor实现与使用Python操作HBase及Coprocessor等实例。主要目的是完善一个简单的实例文档。

## 安装JPype

使用Python操作HBase、Coprocessor应首先安装Python、HBase、JPype，其中Python及HBase的安装这里不再赘述，主要讲解一下JPype的安装情况。（linux）

* 下载JPype安装包，并解压
  + Wget <http://sourceforge.net/projects/jpype/files/JPype/0.5.4/JPype-0.5.4.2.zip>
  + Unzip JPype-0.5.4.2.zip
* 安装g++与python-dev，Python-dev是python的一个静态库与开发工具
  + sudo apt-get install g++ python-dev。
* 进入JPype目录并安装JPype
  + python setup.py install

## Coprocessor

Coprocessor是HBase的一个工具，在这里只关心Coprocessor的安装使用、简单计数的具体实现。

### Coprocessor安装

安装Coprocessor需要首先安装HBase、完成Coprocessor代码。在这里假设已经完成Coprocessor的程序，直接使用。

* 导出Coprocessor的jar包，设为CoprocessorDemo.jar
* 进入HBase安装包，编辑hbase-env.sh，加入：
  + Export HBASE\_CLASSPATH=”../../../CoprocessorDemo.jar”
* 进入HBase安装包，编辑hbase-site.xml，加入：
  + <property>  
    <name>hbase.coprocessor.region.classes</name>  
    <value>hbase.coprocessor.RowCountEndpoint</value>  
    <description>value就是具体的Coprocessor的路径</description>  
    </property>
* 启动HBase即可

### Coprocessor的具体实现

本人使用Coprocessor完成了简单的功能，这里举一个count的例子，主要是Coprocessor的简单应用。

* 继承类BaseEndpointCoprocessor
  + Public class RowCountEndpoint extends BaseEndpointCoprocessor
* 完成count功能
  + 代码见附录1
* 使用Coprocessor的count功能
  + 代码见附录2

## Python操作HBase及Coprocessor

Python操作HBase是通过thrift，而Python操作Coprocessor是通过JPype。

### Python操作HBase

Python操作HBase主要是通过thrift方式，python调用thrift提供的类及方法才创建删除表、插入删除数据等操作。

* 安装thrift
* 进入HBase目录，找到hbase.thrift，生成thrift代码
  + Thrift –gen py hbase.thfirt
  + 生成一个文件夹gen-py，通过操作该文件夹里面的类和方法操作HBase
* 进入HBase目录，启动thrit服务器，设置监听端口为19090
  + Bin/hbase thrift –p 19090 start
* Python使用thrift操作HBase
  + 导入包并将依赖包添加到path  
    Import sys  
    sys.path.append(‘../gen-py’) ‘’’生成的gen-py目录地址  
    form thrift import Thrift  
    from thrift.transport import TSocket  
    from thrift.transport import TTransport  
    from thrift.protocol import TBinaryProtocol  
    from hbase import HBase  
    from hbase.ttypes import \*
  + 创建监听对象  
    transport=TSocket.TSocket(‘localhost’, 19090)  
    transport=TTransprot.TBufferedTransport(transport)  
    protocol=TBinaryProtocol.TBinaryProtocol(transport)  
    client=Hbase.Client(protocol)
  + 打开端口并操作HBase  
    transport.open()  
    ‘’’打印出所有的table名字’’’  
    print ‘table name:’, client.getTableNames()  
    ‘’’删除table’’’  
    client.disableTable(‘tablename’)  
    client.deleteTable(‘tablename’)

### Python操作Coprocessor

Coprocessor是使用Java完成，如果想要Python操作Coprocessor，需要通过使用JPype来操作Java。

* 导入依赖包并将jar包位置加入到path  
  Import jpype  
  import os.path  
  jarPath=os.path.join(os.path.abspath(‘.’), ‘/home/../py’) ‘’’jar包放在py下面  
  jvmPath=jpype.getDefaultJVMPath()
* 启动Java虚拟机  
  jpype.startJVM(jvmPath, “-Djava.ext.dirs=%s” % jarPath)
* 创建Java对象及调用Java方法  
  Example=jpype.JClass(‘common/product/ProductExample’)  
  exampleInstance=Example()  
  exampleInstance.getCount(‘tablename’)  
  system=jpype.JClass(‘java.lang.System’)  
  system.out.println(‘hello world’)
* 关闭Java虚拟机  
  jpype.shutdownJVM()

## 附录

### Coprocessor实现Count代码

public long getCount() throws IOException {

Scan scan = new Scan();

scan.setMaxVersions(1);

RegionCoprocessorEnvironment environment = (RegionCoprocessorEnvironment) getEnvironment();

// use an internal scanner to perform scanning.

InternalScanner scanner = environment.getRegion().getScanner(scan);

int result = 0;

try {

List<KeyValue> curVals = new ArrayList<KeyValue>();

boolean done = false;

do {

curVals.clear();

done = scanner.next(curVals);

result ++;

} while (done);

} finally {

scanner.close();

}

return result;

}

### Coprocessor使用

public static void getCount(String tableName) throws IOException{

HTable table = new HTable(conf, tableName);

try {

Map<byte[], Long> results = table.coprocessorExec(

RowCountProtocol.class, null, null,

new Batch.Call<RowCountProtocol, Long>() {

@Override

public Long call(RowCountProtocol counter)

throws IOException {

return counter.getRowCount();

}

});

long total=0;

for (Map.Entry<byte[], Long> entry : results.entrySet()) {

System.out.println("Region: " + Bytes.toString(entry.getKey()));

total=total+entry.getValue();

}

System.out.println("total count:"+total);

} catch (Throwable throwable) {

throwable.printStackTrace();

}

}