

腾讯云列式数据库HBase

快速入门

产品文档



腾讯云

【版权声明】

©2013-2017 腾讯云版权所有

本文档著作权归腾讯云单独所有，未经腾讯云事先书面许可，任何主体不得以任何形式复制、修改、抄袭、传播全部或部分本文档内容。

【商标声明】



及其它腾讯云服务相关的商标均为腾讯云计算（北京）有限责任公司及其关联公司所有。本文档涉及的第三方主体的商标，依法由权利人所有。

【服务声明】

本文档意在向客户介绍腾讯云全部或部分产品、服务的当时的整体概况，部分产品、服务的内容可能有所调整。您所购买的腾讯云产品、服务的种类、服务标准等应由您与腾讯云之间的商业合同约定，除非双方另有约定，否则，腾讯云对本文档内容不做任何明示或模式的承诺或保证。

文档目录

文档声明.....	2
快速入门.....	4
Step1.连接和访问.....	4
Step2.替换客户端jar包	6
Step3.连接腾讯云参数设置	17
Step4.操作示例代码	18
MapReduce示例代码.....	27
endpoint协处理器示例代码.....	32
observer协处理器示例.....	43
支持SQL查询	44
各语言支持	46

快速入门

Step1.连接和访问

获取连接IP

1) 获取实例ID和IP信息

进入腾讯云Hbase管理中心，可以查看实例ID，并获取连接Hbase的一个或多个IP和端口：

云存储Hbase-实例列表 全部项目 ∨ 华南地区（广州） ∨

分配至项目

☐ 实例ID ∨	所属项目	所属地域	可用区	网络	连接IP	创建时间 ∨	管理
☐ chb-lpvsvdlr	默认项目	华南地区（广州）	广州一区	基础网络	10.66.133.178:2182	2016-06-12 10:11	详情

通过SHELL访问

首先[下载](#)Hbase客户端软件，然后解压到云主机，然后修改conf下的hbase-site.xml添加如下配置项目：

```
<configuration>
  <property>
    <name>hbase.zookeeper.quorum</name>
    <value>(腾讯云提供的连接地址和端口，管理控制台可查)</value>
  </property>
  <property>
    <name> chbase.tencent.enable </name>
    <value> true</value>
  </property>
</configuration>
```

然后执行bin/hbase shell，可以进入Hbase命令终端：

```
ubuntu@VM-26-0-ubuntu: ~/hbase-test/hbase-1.1.3$ bin/hbase shell
2016-06-29 13:55:01,833 WARN [main] util.NativeCodeLoader: Unable to load native-hadoop
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.1.3, r199759, Mon Jun 27 15:21:38 2016

hbase(main):001:0> list
TABLE
monitordata_201603
test
2 row(s) in 0.4670 seconds

=> ["monitordata_201603", "test"]
hbase(main):002:0>
```

Step2.替换客户端jar包

替换客户端jar包

连接腾讯云Hbase服务，需要使用我们提供的jar包来替换社区版本的部分jar包，并设置相应的参数

1).如您使用maven管理jar包，可参考如下pom设置来拉取和替换jar包：

```
<properties>
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  <hadoop-two.version>2.6.4</hadoop-two.version>
</properties>

<dependencies>
  <!-- 导入社区jar包 -->
  <dependency>
    <groupId>org.apache.hbase</groupId>
    <artifactId>hbase-client</artifactId>
    <version>1.1.3</version>
    <type>pom</type>
    <scope>provided</scope>
    <exclusions>
      <exclusion>
        <groupId>org.apache.hadoop</groupId>
        <artifactId>hadoop-annotations</artifactId>
      </exclusion>
      <exclusion>
        <groupId>org.apache.hadoop</groupId>
        <artifactId>hadoop-auth</artifactId>
      </exclusion>
      <exclusion>
        <groupId>org.apache.hadoop</groupId>
        <artifactId>hadoop-client</artifactId>
      </exclusion>
    </exclusions>
  </dependency>
</dependencies>
```

```
<exclusion>
<groupId>org.apache.hadoop</groupId>
<artifactId>hadoop-common</artifactId>
</exclusion>
<exclusion>
<groupId>org.apache.hadoop</groupId>
<artifactId>hadoop-hdfs</artifactId>
</exclusion>
<exclusion>
<groupId>org.apache.hadoop</groupId>
<artifactId>hadoop-mapreduce-client-app</artifactId>
</exclusion>
<exclusion>
<groupId>org.apache.hadoop</groupId>
<artifactId>hadoop-mapreduce-client-common</artifactId>
</exclusion>
<exclusion>
<groupId>org.apache.hadoop</groupId>
<artifactId>hadoop-mapreduce-client-core</artifactId>
</exclusion>
<exclusion>
<groupId>org.apache.hadoop</groupId>
<artifactId>hadoop-mapreduce-client-jobclient</artifactId>
</exclusion>
<exclusion>
<groupId>org.apache.hadoop</groupId>
<artifactId>hadoop-mapreduce-client-shuffle</artifactId>
</exclusion>
<exclusion>
<groupId>org.apache.hadoop</groupId>
<artifactId>hadoop-yarn-api</artifactId>
</exclusion>
<exclusion>
<groupId>org.apache.hadoop</groupId>
```

```
<artifactId>hadoop-yarn-client</artifactId>
</exclusion>
<exclusion>
<groupId>org.apache.hadoop</groupId>
<artifactId>hadoop-yarn-common</artifactId>
</exclusion>
<exclusion>
<groupId>org.apache.hadoop</groupId>
<artifactId>hadoop-yarn-server-common</artifactId>
</exclusion>
</exclusions>
</dependency>
<dependency>
<groupId>org.apache.hbase</groupId>
<artifactId>hbase-server</artifactId>
<version>1.1.3</version>
<type>pom</type>
<scope>provided</scope>
<exclusions>
<exclusion>
<groupId>org.apache.hadoop</groupId>
<artifactId>hadoop-annotations</artifactId>
</exclusion>
<exclusion>
<groupId>org.apache.hadoop</groupId>
<artifactId>hadoop-auth</artifactId>
</exclusion>
<exclusion>
<groupId>org.apache.hadoop</groupId>
<artifactId>hadoop-client</artifactId>
</exclusion>
<exclusion>
<groupId>org.apache.hadoop</groupId>
<artifactId>hadoop-common</artifactId>
```

```
</exclusion>
<exclusion>
<groupId>org.apache.hadoop</groupId>
<artifactId>hadoop-hdfs</artifactId>
</exclusion>
<exclusion>
<groupId>org.apache.hadoop</groupId>
<artifactId>hadoop-mapreduce-client-app</artifactId>
</exclusion>
<exclusion>
<groupId>org.apache.hadoop</groupId>
<artifactId>hadoop-mapreduce-client-common</artifactId>
</exclusion>
<exclusion>
<groupId>org.apache.hadoop</groupId>
<artifactId>hadoop-mapreduce-client-core</artifactId>
</exclusion>
<exclusion>
<groupId>org.apache.hadoop</groupId>
<artifactId>hadoop-mapreduce-client-jobclient</artifactId>
</exclusion>
<exclusion>
<groupId>org.apache.hadoop</groupId>
<artifactId>hadoop-mapreduce-client-shuffle</artifactId>
</exclusion>
<exclusion>
<groupId>org.apache.hadoop</groupId>
<artifactId>hadoop-yarn-api</artifactId>
</exclusion>
<exclusion>
<groupId>org.apache.hadoop</groupId>
<artifactId>hadoop-yarn-client</artifactId>
</exclusion>
<exclusion>
```

```
<groupId>org.apache.hadoop</groupId>
<artifactId>hadoop-yarn-common</artifactId>
</exclusion>
<exclusion>
<groupId>org.apache.hadoop</groupId>
<artifactId>hadoop-yarn-server-common</artifactId>
</exclusion>
</exclusions>
</dependency>
<dependency>
<groupId>com.google.code.gson</groupId>
<artifactId>gson</artifactId>
<version>2.5</version>
</dependency>
<dependency>
<groupId>commons-configuration</groupId>
<artifactId>commons-configuration</artifactId>
<version>1.6</version>
</dependency>
<dependency>
<groupId>org.apache.httpcomponents</groupId>
<artifactId>httpcore</artifactId>
<version>4.2.4</version>
</dependency>
<dependency>
<groupId>org.apache.httpcomponents</groupId>
<artifactId>httpclient</artifactId>
<version>4.2.5</version>
</dependency>
<dependency>
<groupId>org.htrace</groupId>
<artifactId>htrace-core</artifactId>
<version>3.0.4</version>
</dependency>
```

```
<dependency>
<groupId>org.apache.avro</groupId>
<artifactId>avro</artifactId>
<version>1.7.4</version>
</dependency>
<!-- 导入腾讯云提供的jar包，注意修改jar包路径 -->
<dependency>
<groupId>org.apache.hadoop</groupId>
<artifactId>hadoop-annotations</artifactId>
<version>2.6.4</version>
<scope>system</scope>
<systemPath>${basedir}/hadoop/hadoop-annotations-2.6.4.jar</systemPath>
</dependency>
<dependency>
<groupId>org.apache.hadoop</groupId>
<artifactId>hadoop-auth</artifactId>
<version>2.6.4</version>
<scope>system</scope>
<systemPath>${basedir}/hadoop/hadoop-auth-2.6.4.jar</systemPath>
</dependency>
<dependency>
<groupId>org.apache.hadoop</groupId>
<artifactId>hadoop-client</artifactId>
<version>2.6.4</version>
<scope>system</scope>
<systemPath>${basedir}/hadoop/hadoop-client-2.6.4.jar</systemPath>
</dependency>
<dependency>
<groupId>org.apache.hadoop</groupId>
<artifactId>hadoop-common</artifactId>
<version>2.6.4</version>
<scope>system</scope>
<systemPath>${basedir}/hadoop/hadoop-common-2.6.4.jar</systemPath>
</dependency>
```

```
<dependency>
<groupId>org.apache.hadoop</groupId>
<artifactId>hadoop-hdfs</artifactId>
<version>2.6.4</version>
<scope>system</scope>
<systemPath>${basedir}/hadoop/hadoop-hdfs-2.6.4.jar</systemPath>
</dependency>
<dependency>
<groupId>org.apache.hadoop</groupId>
<artifactId>hadoop-mapreduce-client-app</artifactId>
<version>2.6.4</version>
<scope>system</scope>
<systemPath>${basedir}/hadoop/hadoop-mapreduce-client-app-2.6.4.jar</systemPath>
</dependency>
<dependency>
<groupId>org.apache.hadoop</groupId>
<artifactId>hadoop-mapreduce-client-common</artifactId>
<version>2.6.4</version>
<scope>system</scope>
<systemPath>${basedir}/hadoop/hadoop-mapreduce-client-common-2.6.4.jar</systemPath>
</dependency>
<dependency>
<groupId>org.apache.hadoop</groupId>
<artifactId>hadoop-mapreduce-client-core</artifactId>
<version>2.6.4</version>
<scope>system</scope>
<systemPath>${basedir}/hadoop/hadoop-mapreduce-client-core-2.6.4.jar</systemPath>
</dependency>
<dependency>
<groupId>org.apache.hadoop</groupId>
<artifactId>hadoop-mapreduce-client-jobclient</artifactId>
<version>2.6.4</version>
<scope>system</scope>
<systemPath>${basedir}/hadoop/hadoop-mapreduce-client-jobclient-2.6.4.jar</systemPath>
```

```
</dependency>
<dependency>
<groupId>org.apache.hadoop</groupId>
<artifactId>hadoop-mapreduce-client-shuffle</artifactId>
<version>2.6.4</version>
<scope>system</scope>
<systemPath>${basedir}/hadoop/hadoop-mapreduce-client-shuffle-2.6.4.jar</systemPath>
</dependency>
<dependency>
<groupId>org.apache.hadoop</groupId>
<artifactId>hadoop-yarn-api</artifactId>
<version>2.6.4</version>
<scope>system</scope>
<systemPath>${basedir}/hadoop/hadoop-yarn-api-2.6.4.jar</systemPath>
</dependency>
<dependency>
<groupId>org.apache.hadoop</groupId>
<artifactId>hadoop-yarn-client</artifactId>
<version>2.6.4</version>
<scope>system</scope>
<systemPath>${basedir}/hadoop/hadoop-yarn-client-2.6.4.jar</systemPath>
</dependency>
<dependency>
<groupId>org.apache.hadoop</groupId>
<artifactId>hadoop-yarn-common</artifactId>
<version>2.6.4</version>
<scope>system</scope>
<systemPath>${basedir}/hadoop/hadoop-yarn-common-2.6.4.jar</systemPath>
</dependency>
<dependency>
<groupId>org.apache.hadoop</groupId>
<artifactId>hadoop-yarn-server-common</artifactId>
<version>2.6.4</version>
<scope>system</scope>
```

```
<systemPath>${basedir}/hadoop/hadoop-yarn-server-common-2.6.4.jar</systemPath>
</dependency>
<dependency>
<groupId>org.apache.hbase</groupId>
<artifactId>hbase-client</artifactId>
<version>1.1.3</version>
<scope>system</scope>
<systemPath>${basedir}/hadoop/hbase-client-1.1.3.jar</systemPath>
</dependency>
<dependency>
<groupId>org.apache.hbase</groupId>
<artifactId>hbase-common</artifactId>
<version>1.1.3</version>
<scope>system</scope>
<systemPath>${basedir}/hadoop/hbase-common-1.1.3.jar</systemPath>
</dependency>
<dependency>
<groupId>org.apache.hbase</groupId>
<artifactId>hbase-protocol</artifactId>
<version>1.1.3</version>
<scope>system</scope>
<systemPath>${basedir}/hadoop/hbase-protocol-1.1.3.jar</systemPath>
</dependency>
<dependency>
<groupId>org.apache.hbase</groupId>
<artifactId>hbase-server</artifactId>
<version>1.1.3</version>
<scope>system</scope>
<systemPath>${basedir}/hadoop/hbase-server-1.1.3.jar</systemPath>
</dependency>
</dependencies>
```

需要替换的jar包[下载](#)

需要替换的jar包清单：

hadoop-annotations-2.5.1.jar
hadoop-auth-2.5.1.jar
hadoop-client-2.5.1.jar
hadoop-common-2.5.1.jar
hadoop-hdfs-2.5.1.jar
hadoop-mapreduce-client-app-2.5.1.jar
hadoop-mapreduce-client-common-2.5.1.jar
hadoop-mapreduce-client-core-2.5.1.jar
hadoop-mapreduce-client-jobclient-2.5.1.jar
hadoop-mapreduce-client-shuffle-2.5.1.jar
hadoop-yarn-api-2.5.1.jar
hadoop-yarn-client-2.5.1.jar
hadoop-yarn-common-2.5.1.jar
hadoop-yarn-server-common-2.5.1.jar
hbase-client-1.1.3.jar
hbase-common-1.1.3.jar
hbase-protocol-1.1.3.jar
hbase-server-1.1.3.jar

以上jar包替换为：

hadoop-annotations-2.6.4.jar
hadoop-auth-2.6.4.jar
hadoop-client-2.6.4.jar
hadoop-common-2.6.4.jar
hadoop-hdfs-2.6.4.jar
hadoop-mapreduce-client-app-2.6.4.jar
hadoop-mapreduce-client-common-2.6.4.jar
hadoop-mapreduce-client-core-2.6.4.jar
hadoop-mapreduce-client-jobclient-2.6.4.jar
hadoop-mapreduce-client-shuffle-2.6.4.jar
hadoop-yarn-api-2.6.4.jar
hadoop-yarn-client-2.6.4.jar
hadoop-yarn-common-2.6.4.jar

hadoop-yarn-server-common-2.6.4.jar

hbase-client-1.1.3.jar

hbase-common-1.1.3.jar

hbase-protocol-1.1.3.jar

hbase-server-1.1.3.jar

2).如不使用mvn获取依赖，您也可以在这里直接[下载](#)所有的客户端jar包

Step3.连接腾讯云参数设置

连接腾讯云参数设置

1).连接腾讯云Hbase服务时必须设置以下参数为true（完整代码请参考示例代码），方能正常使用：

```
config.setBoolean("chbase.tencent.enable", true);
```

2).如需要使用yarn，还需要额外设置实例ID（管理页面可以查到，完整代码请参考示例代码），如：

```
config.set("yarn.chbase.tencent.instanceid", "chb-lpvsvdlr");
```

提示：

除增加以上代码，其余使用方式和社区版本Hbase一致；可以参考<https://hbase.apache.org/> 的API文档；

如您未设置以上参数，也可以正常连接社区版的Hbase。

Step4.操作示例代码

参考以下示例代码，包括常见的hbase表格创建、删除，插入、删除、读取数据等操作

```
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.Map;
import java.util.NavigableMap;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.hbase.Cell;
import org.apache.hadoop.hbase.CellUtil;
import org.apache.hadoop.hbase.HBaseConfiguration;
import org.apache.hadoop.hbase.HColumnDescriptor;
import org.apache.hadoop.hbase.HTableDescriptor;
import org.apache.hadoop.hbase.KeyValue;
import org.apache.hadoop.hbase.TableName;
import org.apache.hadoop.hbase.client.Admin;
import org.apache.hadoop.hbase.client.Connection;
import org.apache.hadoop.hbase.client.ConnectionFactory;
import org.apache.hadoop.hbase.client.Delete;
import org.apache.hadoop.hbase.client.Get;
import org.apache.hadoop.hbase.client.Put;
import org.apache.hadoop.hbase.client.Result;
import org.apache.hadoop.hbase.client.ResultScanner;
import org.apache.hadoop.hbase.client.Scan;
import org.apache.hadoop.hbase.client.Table;
import org.apache.hadoop.hbase.util.Bytes;

public class Connect {

    public static void createTable(String tableName, String[] familys) throws IOException {
        Admin admin = null;
        Connection con = null;
```

```
try{
Configuration config = HBaseConfiguration.create();
// 填写zookeeper地址，多个地址用英文逗号隔开
config.set("hbase.zookeeper.quorum", "10.66.133.178:2181");
// 设置重试参数
config.setInt("hbase.client.retries.number", 1);
/*
* 要连接腾讯云的hbase服务必须设置此值为true；不设置该值功能和社区版相同，可以正常连接自建hbase
服务
*/
config.setBoolean("chbase.tencent.enable", true);

TableName TABLE = TableName.valueOf(tableName);
con = ConnectionFactory.createConnection(config);
admin = con.getAdmin();

if (admin.tableExists(TABLE)) {
System.out.println("table already exists!");
} else {
HTableDescriptor tableDesc = new HTableDescriptor(TABLE);
for (int i = 0; i < familys.length; i++) {
tableDesc.addFamily(new HColumnDescriptor(familys[i]));
}
admin.createTable(tableDesc);
System.out.println("create table " + tableName + " ok.");
}
}catch(IOException e){
e.printStackTrace();
}finally{
admin.close();
con.close();
}
}
```

```
public static void deleteTable(String tableName) throws IOException{
    Admin admin = null;
    Connection con = null;
    try{
        Configuration config = HBaseConfiguration.create();
        // 填写zookeeper地址，多个地址用英文逗号隔开
        config.set("hbase.zookeeper.quorum", "10.66.133.178:2181");
        // 设置重试参数
        config.setInt("hbase.client.retries.number", 1);
        /*
        * 要连接腾讯云的hbase服务必须设置此值为true；不设置该值功能和社区版相同，可以正常连接自建hbase
        服务
        */
        config.setBoolean("chbase.tencent.enable", true);

        TableName TABLE = TableName.valueOf(tableName);
        con = ConnectionFactory.createConnection(config);
        admin = con.getAdmin();

        if (!admin.tableExists(TABLE)) {
            System.out.println("table not exists!");
        } else {
            admin.disableTable(TABLE);
            admin.deleteTable(TABLE);
            System.out.println("delete table " + tableName + " ok.");
        }
    }catch(IOException e){
        e.printStackTrace();
    }finally{
        admin.close();
        con.close();
    }
}
```

```
public static void get(String tablename, String rowkey) throws IOException {
    Connection con = null;
    Table table = null;
    try{
        Configuration config = HBaseConfiguration.create();
        // 填写zookeeper地址，多个地址用英文逗号隔开
        config.set("hbase.zookeeper.quorum", "10.66.133.178:2181");
        // 设置重试参数
        config.setInt("hbase.client.retries.number", 1);
        /*
        * 要连接腾讯云的hbase服务必须设置此值为true；不设置该值功能和社区版相同，可以正常连接自建hbase
        服务
        */
        config.setBoolean("chbase.tencent.enable", true);

        con = ConnectionFactory.createConnection(config);
        table = con.getTable(TableName.valueOf(tablename));

        Get get = new Get(rowkey.getBytes());
        Result rs = table.get(get);
        for (Cell cell : rs.rawCells()) {
            System.out.print(new String(CellUtil.cloneRow(cell)) + " ");
            System.out.print(new String(CellUtil.cloneFamily(cell)) + ":");
            System.out.print(new String(CellUtil.cloneQualifier(cell)) + " ");
            System.out.print(cell.getTimestamp() + " ");
            System.out.println(new String(CellUtil.cloneValue(cell)));
        }
    }catch(IOException e){
        e.printStackTrace();
    }finally{
        table.close();
        con.close();
    }
}
```

```
public static void del(String tablename, String rowkey) throws IOException {
    Connection con = null;
    Table table = null;
    try{
        Configuration config = HBaseConfiguration.create();
        // 填写zookeeper地址，多个地址用英文逗号隔开
        config.set("hbase.zookeeper.quorum", "10.66.133.178:2181");
        // 设置重试参数
        config.setInt("hbase.client.retries.number", 1);
        /*
        * 要连接腾讯云的hbase服务必须设置此值为true；不设置该值功能和社区版相同，可以正常连接自建hbase
        服务
        */
        config.setBoolean("chbase.tencent.enable", true);

        con = ConnectionFactory.createConnection(config);
        table = con.getTable(TableName.valueOf(tablename));
        // 批量删除
        List<Delete> list = new ArrayList<Delete>();
        Delete del = new Delete(rowkey.getBytes());
        list.add(del);
        table.delete(list);
        // 单个删除
        // Delete del = new Delete(Bytes.toBytes(rowkey));
        // table.delete(del);
        System.out.println("del recored " + rowkey + " ok.");
    }catch(IOException e){
        e.printStackTrace();
    }finally{
        table.close();
        con.close();
    }
}
```

```
public static void put(String tablename, String rowkey, String familyname, String colname, String
value) throws IOException
{
    Connection con = null;
    Table table = null;
    try{
        Configuration config = HBaseConfiguration.create();
        // 填写zookeeper地址，多个地址用英文逗号隔开
        config.set("hbase.zookeeper.quorum", "10.66.133.178:2181");
        // 设置重试参数
        config.setInt("hbase.client.retries.number", 1);
        /*
        * 要连接腾讯云的hbase服务必须设置此值为true；不设置该值功能和社区版相同，可以正常连接自建hbase
        服务
        */
        config.setBoolean("chbase.tencent.enable", true);

        con = ConnectionFactory.createConnection(config);
        table = con.getTable(TableName.valueOf(tablename));

        byte[] ROWKEY = Bytes.toBytes(rowkey);
        Put put = new Put(ROWKEY);

        Cell c1 = CellUtil.createCell(ROWKEY, Bytes.toBytes(familyname), Bytes.toBytes(colname),
        System.currentTimeMillis(), KeyValue.Type.Put.getCode(), Bytes.toBytes(value));
        put.add(c1);
        table.put(put);
        System.out.println("-----put ok-----");
    }catch(IOException e){
        e.printStackTrace();
    }finally{
        table.close();
        con.close();
    }
}
```

```
}  
}  
  
public static void scan(String tablename, String rowkey, String family) throws IOException {  
    Connection con = null;  
    Table table = null;  
    try{  
        Configuration config = HBaseConfiguration.create();  
        // 填写zookeeper地址，多个地址用英文逗号隔开  
        config.set("hbase.zookeeper.quorum", "10.66.133.178:2181");  
        // 设置重试参数  
        config.setInt("hbase.client.retries.number", 1);  
        /*  
        * 要连接腾讯云的hbase服务必须设置此值为true；不设置该值功能和社区版相同，可以正常连接自建hbase  
        服务  
        */  
        config.setBoolean("chbase.tencent.enable", true);  
        Scan scan = new Scan();  
        scan.setStartRow(Bytes.toBytes(rowkey));  
        scan.setCaching(500);  
        scan.setCacheBlocks(false);  
  
        con = ConnectionFactory.createConnection(config);  
        table = con.getTable(TableName.valueOf(tablename));  
  
        ResultScanner ss = table.getScanner(scan);  
        System.out.println("-----");  
        for (Result r : ss) {  
            NavigableMap<byte[], byte[]> map = r.getFamilyMap(Bytes.toBytes(family));  
            for (Map.Entry<byte[], byte[]> ent : map.entrySet()) {  
                String key = new String(ent.getKey());  
                String value = new String(ent.getValue());  
                System.out.println(  
                    "find result is:" + new String(r.getRow()) + " and code is:" + key + " and value is:" + value);  
            }  
        }  
    }  
}
```

```
}  
}  
System.out.println("-----");  
}catch(IOException e){  
    e.printStackTrace();  
}finally{  
    table.close();  
    con.close();  
}  
}
```

```
public static void main(String[] args) throws Exception {  
    if (args == null || args.length < 1) {  
        System.out.println("please input args....");  
        return;  
    }  
    String op = args[0];
```

```
    if (op.equals("create")) {  
        // 要创建的列名  
        String[] familys = { "fam1", "fam2", "fam3" };  
        createTable("test", familys);  
    } else if (op.equals("put")) {  
        put("test", "key1", "fam1", "col1", "value1");  
    } else if (op.equals("get")) {  
        get("test", "key1");  
    } else if (op.equals("deltable")) {  
        deleteTable("test");  
    } else if (op.equals("scan")) {  
        scan("test", "key1", "fam1");  
    } else if (op.equals("del")) {  
        del("test", "key1");  
    }  
}
```

```
}
```

MapReduce示例代码

Map类

```
import java.io.IOException;
import java.util.Map;
import java.util.NavigableMap;

import org.apache.hadoop.hbase.client.Result;
import org.apache.hadoop.hbase.io.ImmutableBytesWritable;
import org.apache.hadoop.hbase.mapreduce.TableMapper;
import org.apache.hadoop.hbase.util.Bytes;
import org.apache.hadoop.io.Text;

public class Mapper extends TableMapper<Text, Text>{
    @Override
    protected void map(ImmutableBytesWritable rowkey, Result columns, Context context)
        throws IOException, InterruptedException{
        NavigableMap<byte[], byte[]> map = columns.getFamilyMap(Bytes.toBytes("retcode"));
        for (Map.Entry<byte[], byte[]> ent : map.entrySet()) {
            String retcode=Bytes.toString(ent.getKey());
            String value = Bytes.toString(ent.getValue());

            Text retkey = new Text(retcode);
            Text retvalue = new Text(value);
            context.write(retkey, retvalue);
        }
    }
}
```

Reduce类

```
import java.io.IOException;
```

```
import org.apache.hadoop.hbase.io.ImmutableBytesWritable;
import org.apache.hadoop.hbase.mapreduce.TableReducer;
import org.apache.hadoop.io.Text;

public class Reduce extends TableReducer<Text, Text, ImmutableBytesWritable> {
    @Override
    protected void cleanup(Context context) throws IOException, InterruptedException {

    }

    @Override
    protected void setup(Context context) throws IOException, InterruptedException {

    }

    super.setup(context);
}

@Override
public void reduce(Text key, Iterable<Text> values, Context context) throws IOException,
InterruptedException {

    long size = 0;
    String rowkey = key.toString();
    System.out.println(rowkey);
    for(Text t:values){
        System.out.println(t.toString());
    }
}
}
```

提交任务

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.hbase.client.Scan;
```

```
import org.apache.hadoop.hbase.mapreduce.TableMapReduceUtil;
import org.apache.hadoop.hbase.util.Bytes;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.output.NullOutputFormat;

public class MR {

    public static void main(String[] args) throws Exception{
        Job job =Job.getInstance();
        Configuration conf=job.getConfiguration();
        // 填写zookeeper地址，多个地址用英文逗号隔开
        conf.set("hbase.zookeeper.quorum", "10.66.133.178:2181");
        //必填：填写腾讯云Hbase实例ID
        conf.set("yarn.chbase.tencent.instanceid", "chb-lpvsvdlr");
        job.setJobName("testjob");

        String tableName = "monitordata_201603";
        Scan scan = new Scan();
        scan.setStartRow(Bytes.toBytes("0800:00_00000000"));
        scan.setCaching(500);
        scan.setCacheBlocks(false);
        job.setJarByClass(MR.class);
        job.setReducerClass(Reduce.class);
        job.setOutputFormatClass(NullOutputFormat.class);
        job.setNumReduceTasks(5);
        //如果要使用第三方jar包，可使用该工具类上传
        //job.addFileToClassPath(JobHelper.addJarToDistributedCache(GenericObjectPoolConfig.class,
        conf));

        TableMapReduceUtil.initTableMapperJob(tableName, scan, Mapper.class, Text.class, Text.class, job);

        boolean b = job.waitForCompletion(true);
        if (!b) {
```

```
throw new Exception("error with job!");  
}  
}
```

工具类（如果需要使用第三方jar包）

```
import java.io.IOException;  
import org.apache.hadoop.fs.Path;  
import org.apache.hadoop.fs.FileSystem;  
import org.apache.hadoop.conf.Configuration;  
import java.io.File;  
  
public class JobHelper {  
  
    public JobHelper() {  
  
    }  
  
    public static Path addJarToDistributedCache(Class classToAdd, Configuration conf) throws  
IOException {  
  
        // Retrieve jar file for class2Add  
        String jar = classToAdd.getProtectionDomain().getCodeSource().getLocation().getPath();  
  
        File jarFile = new File(jar);  
  
        // Declare new HDFS location  
        Path hdfsJar = new Path("/tmp/hadoop/userlib/" + jarFile.getName());  
  
        // Mount HDFS  
        FileSystem hdfs = FileSystem.get(conf);  
  
        // Copy (override) jar file to HDFS
```

```
hdfs.copyFromLocalFile(false, true, new Path(jar), hdfsJar);  
hdfs.close();  
return hdfsJar;  
}  
}
```

endpoint协处理器示例代码

使用协处理，需要将jar包提供给我们，我们安装后提供hdfs地址，您可以再通过shell或者api方式（代码中有示例）安装。强烈建议开发好协处理器jar包后，经过充分测试再安装，避免安装后影响Hbase服务的正常使用。

协处理器开发步骤

- 1.编写proto文件（示例：RowCount.proto）；因为hbase使用google的protoc-2.5.0版本，所以最好使用相同版本编译proto文件，生成java文件（示例：RowCountService.java）；
- 2.编写服务端EndPoint代码（示例：RowCountEndPoint.java）；
- 3.编写客户端调用Client代码（示例：RowCountClient.java）；
- 4.编译jar包

endpoint协处理器示例：

proto文件

```
option java_package = "com.tencent.yun.endpoint.proto";
option java_outer_classname = "RowCountService";
option java_generic_services = true;
option java_generate_equals_and_hash = true;
option optimize_for = SPEED;
```

```
message RowCountRequest{
    required string family = 1;
    required string column = 2;
}
```

```
message RowCountResponse {
    required int64 rowCount = 1 [default = 0];
}
```

```
service RowCount {
```

```
rpc getRowCount(RowCountRequest)
returns (RowCountResponse);
}
```

服务端RowCountEndPoint类

```
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

import org.apache.hadoop.hbase.Cell;
import org.apache.hadoop.hbase.Coprocessor;
import org.apache.hadoop.hbase.CoprocessorEnvironment;
import org.apache.hadoop.hbase.client.Scan;
import org.apache.hadoop.hbase.coprocessor.CoprocessorException;
import org.apache.hadoop.hbase.coprocessor.CoprocessorService;
import org.apache.hadoop.hbase.coprocessor.RegionCoprocessorEnvironment;
import org.apache.hadoop.hbase.protobuf.ResponseConverter;
import org.apache.hadoop.hbase.regionserver.InternalScanner;
import org.apache.hadoop.hbase.util.Bytes;

import com.google.protobuf.RpcCallback;
import com.google.protobuf.RpcController;
import com.google.protobuf.Service;
import com.tencent.yun.endpoint.proto.RowCountService;
import com.tencent.yun.endpoint.proto.RowCountService.RowCountRequest;
import com.tencent.yun.endpoint.proto.RowCountService.RowCountResponse;

public class RowCountEndPoint extends RowCountService.RowCount implements Coprocessor,
CoprocessorService {

    private RegionCoprocessorEnvironment env;
```

```
public Service getService() {  
    return this;  
}
```

```
public void start(CoprocessorEnvironment env) throws IOException {  
    if (env instanceof RegionCoprocessorEnvironment) {  
        this.env = (RegionCoprocessorEnvironment) env;  
    } else {  
        throw new CoprocessorException("Must be loaded on a table region!");  
    }  
}
```

```
public void stop(CoprocessorEnvironment arg0) throws IOException {  
    // do nothing  
  
}
```

```
@Override  
public void getRowCount(RpcController controller, RowCountRequest request,  
    RpcCallback<RowCountResponse> done) {  
    Scan scan = new Scan();  
    scan.addFamily(Bytes.toBytes(request.getFamily()));  
    scan.addColumn(Bytes.toBytes(request.getFamily()), Bytes.toBytes(request.getColumn()));  
    // scan.setMaxVersions(1);  
    InternalScanner scanner = null;  
    RowCountResponse response = null;  
  
    long count = 0L;  
    try {  
        List<Cell> results = new ArrayList<Cell>();  
        boolean hasMore = false;  
        scanner = env.getRegion().getScanner(scan);  
  
        do {
```

```
hasMore = scanner.next(results);
for (Cell cell : results) {
    count++;
    // count = count + Bytes.toLong(CellUtil.cloneValue(cell));
}
results.clear();
} while (hasMore);

response = RowCountResponse.newBuilder().setRowCount(count).build();

} catch (IOException e) {
    ResponseConverter.setControllerException(controller, e);
} finally {
    if (scanner != null) {
        try {
            scanner.close();
        } catch (IOException ignored) {
        }
    }
}
done.run(response);
}

}
```

客户端RowCountClient类

```
import java.io.IOException;
import java.util.Map;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.hbase.Coprocessor;
```

```
import org.apache.hadoop.hbase.HBaseConfiguration;
import org.apache.hadoop.hbase.HColumnDescriptor;
import org.apache.hadoop.hbase.HTableDescriptor;
import org.apache.hadoop.hbase.TableName;
import org.apache.hadoop.hbase.client.Admin;
import org.apache.hadoop.hbase.client.Connection;
import org.apache.hadoop.hbase.client.ConnectionFactory;
import org.apache.hadoop.hbase.client.Table;
import org.apache.hadoop.hbase.client.coprocessor.Batch;
import org.apache.hadoop.hbase.ipc.BlockingRpcCallback;
import org.apache.hadoop.hbase.ipc.ServerRpcController;

import com.google.protobuf.ServiceException;
import com.tencent.yun.endpoint.proto.RowCountService.RowCount;
import com.tencent.yun.endpoint.proto.RowCountService.RowCountRequest;
import com.tencent.yun.endpoint.proto.RowCountService.RowCountResponse;
import com.tencent.yun.observer.RegionObserver;

public class RowCountClient {

    public static void testRowCountEndpoint(String tableName, String family, String col) throws
IOException {
        System.out.println("begin test.....");
        long t1 = System.currentTimeMillis();
        Configuration config = HBaseConfiguration.create();
        //填写hbase zk地址
        config.set("hbase.zookeeper.quorum",
"100.67.159.134:2181,100.67.159.141:2181,100.67.159.196:2181");

        // 填寫family名和列名
        final RowCountRequest req =
RowCountRequest.newBuilder().setFamily(family).setColumn(col).build();
        RowCountResponse resp = null;
        Connection con = null;
```

```
Table table = null;

try {
    con = ConnectionFactory.createConnection(config);
    table = con.getTable(TableName.valueOf(tableName));
    Map<byte[], Long> results = table.coprocessorService(RowCount.class, null, null,
        new Batch.Call<RowCount, Long>() {

        public Long call(RowCount instance) throws IOException {
            ServerRpcController controller = new ServerRpcController();
            BlockingRpcCallback<RowCountResponse> rpccall = new
            BlockingRpcCallback<RowCountResponse>();
            instance.getRowCount(controller, req, rpccall);
            RowCountResponse resp = rpccall.get();
            //
            return resp.hasRowCount() ? resp.getRowCount() : 0L;
        }

    });
    long count = 0L;
    for (Long sum : results.values()) {
        System.out.println("region row Sum = " + sum);
        count += sum;
    }
    System.out.println("total count = " + count);
    long t2 = System.currentTimeMillis();
    System.out.println("use time = " + (t2-t1));
} catch (IOException e) {
    e.printStackTrace();
} catch (ServiceException e) {
    e.printStackTrace();
} catch (Throwable e) {
    e.printStackTrace();
} finally{
    table.close();
}
```

```
con.close();
}
}

public static void delCorprocessor(String tableName) throws IOException {
    System.out.println("begin delCorprocessor.....");
    Connection con = null;
    Admin admin = null;
    try {
        Configuration config = HBaseConfiguration.create();
        //填写hbase zk地址
        config.set("hbase.zookeeper.quorum",
"100.67.159.134:2181,100.67.159.141:2181,100.67.159.196:2181");

        TableName TABLE = TableName.valueOf(tableName);

        con = ConnectionFactory.createConnection(config);
        admin = con.getAdmin();

        HTableDescriptor tableDesc = admin.getTableDescriptor(TABLE);

        tableDesc.removeCoprocessor(RowCountEndPoint.class.getCanonicalName());
        tableDesc.removeCoprocessor(RegionObserver.class.getCanonicalName());

        admin.modifyTable(TABLE, tableDesc);
    } catch (IOException e) {
        e.printStackTrace();
    }finally{
        admin.close();
        con.close();
    }
    System.out.println("end delCorprocessor.....ok");
}
```

```
/**
 * 支持hbase0.96以上版本
 * @throws IOException
 */
public static void setupToExistTable(String tableName) throws IOException {
    System.out.println("begin setupToExistTable.....");
    Connection con = null;
    Admin admin = null;
    try {
        Configuration config = HBaseConfiguration.create();
        //填写hbase zk地址
        config.set("hbase.zookeeper.quorum",
"100.67.159.134:2181,100.67.159.141:2181,100.67.159.196:2181");

        TableName TABLE = TableName.valueOf(tableName);
        con = ConnectionFactory.createConnection(config);
        admin = con.getAdmin();

        HTableDescriptor tableDesc = admin.getTableDescriptor(TABLE);

        //填写我们提供的jar包的hdfs地址
        Path jarPath = new Path("hdfs://100.67.159.132:8020/coprocessor/1/thbase-1.0-SNAPSHOT.jar");

        tableDesc.addCoprocessor(RowCountEndPoint.class.getCanonicalName(), jarPath,
Coprocessor.PRIORITY_USER,
        null);
        tableDesc.addCoprocessor(RegionObserver.class.getCanonicalName(), jarPath,
Coprocessor.PRIORITY_USER, null);

        admin.modifyTable(TABLE, tableDesc);
    } catch (IOException e) {
        e.printStackTrace();
    } finally{
        admin.close();
    }
}
```

```
con.close();
}
System.out.println("end setupToExistTable.....ok");

}

public static void createAndSetup(String tableName) throws IOException {
    System.out.println("begin safesetup.....");
    Connection con = null;
    Admin admin = null;
    try {
        Configuration config = HBaseConfiguration.create();
        //填写hbase zk地址
        config.set("hbase.zookeeper.quorum",
"100.67.159.134:2181,100.67.159.141:2181,100.67.159.196:2181");

        TableName TABLE = TableName.valueOf(tableName);

        con = ConnectionFactory.createConnection(config);
        admin = con.getAdmin();

        HTableDescriptor tableDesc = new HTableDescriptor(TABLE);
        HColumnDescriptor columnFamily1 = new HColumnDescriptor("f1");
        columnFamily1.setMaxVersions(3);
        tableDesc.addFamily(columnFamily1);

        //填写我们提供的jar包的hdfs地址
        Path jarPath = new Path("hdfs://100.67.159.132:8020/coprocessor/1/thbase-1.0-SNAPSHOT.jar");

        tableDesc.addCoprocessor(RowCountEndPoint.class.getCanonicalName(), jarPath,
Coprocessor.PRIORITY_USER,
        null);

        tableDesc.addCoprocessor(RegionObserver.class.getCanonicalName(), jarPath,
Coprocessor.PRIORITY_USER, null);
```

```
admin.createTable(tableDesc);
System.out.println("end safesetup.....ok");
} catch (IOException e) {
    e.printStackTrace();
}finally{
    admin.close();
    con.close();
}

}

public static void main(String[] args) throws IOException {
    if (args == null || args.length < 2) {
        System.out.println("please input args....");
        return;
    }
    String op = args[0];
    String tableName = args[1];
    String family = args[2];
    String col = args[3];

    if (op.equals("setup")) {
        setupToExistTable(tableName);
    } else if (op.equals("safesetup")) {
        createAndSetup(tableName);
    } else if (op.equals("run")) {
        testRowCountEndpoint(tableName, family, col);
    } else if (op.equals("unset")) {
        delCorprocessor(tableName);
    }

}
```

```
}
```

observer协处理器示例

observer协处理器示例：

```
import java.io.IOException;
import java.util.List;

import org.apache.hadoop.hbase.Cell;
import org.apache.hadoop.hbase.CellUtil;
import org.apache.hadoop.hbase.client.Get;
import org.apache.hadoop.hbase.coprocessor.BaseRegionObserver;
import org.apache.hadoop.hbase.coprocessor.ObserverContext;
import org.apache.hadoop.hbase.coprocessor.RegionCoprocessorEnvironment;
import org.apache.hadoop.hbase.util.Bytes;

public class RegionObserver extends BaseRegionObserver {
    private static final byte[] ADMIN = Bytes.toBytes("admin");
    private static final byte[] COLUMN_FAMILY = Bytes.toBytes("f1");
    private static final byte[] COLUMN = Bytes.toBytes("col1");
    private static final byte[] VALUE = Bytes.toBytes("You can not see Admin details");

    @Override
    public void preGetOp(ObserverContext<RegionCoprocessorEnvironment> e, Get get, List<Cell>
results) throws IOException{

        if (Bytes.equals(get.getRow(),ADMIN)) {
            Cell c = CellUtil.createCell(get.getRow(), COLUMN_FAMILY, COLUMN, System.currentTimeMillis(),
(byte)4, VALUE);
            results.add(c);
            e.bypass();
        }

    }
}
```

支持SQL查询

1.phoenix配置

需要使用phoenix需要先下载我们提供的[phoenix的jar包](#)，下载后解压到CVM的任意一个目录下：

```
ubuntu@VM-203-25-ubuntu:~/test/phoenix-4.8.1-HBase-1.1$ ll
total 205264
drwxrwxr-x 7 ubuntu ubuntu 4096 Dec 1 20:46 ./
drwxrwxr-x 5 ubuntu ubuntu 4096 Dec 1 20:46 ../
drwxr-xr-x 2 ubuntu ubuntu 4096 Dec 1 20:57 bin/
-rw-r--r-- 1 ubuntu ubuntu 1930 Sep 23 08:57 build.txt
drwxr-xr-x 3 ubuntu ubuntu 4096 Oct 19 14:33 dev/
drwxr-xr-x 2 ubuntu ubuntu 4096 Oct 19 14:33 docs/
drwxr-xr-x 3 ubuntu ubuntu 4096 Oct 19 14:33 examples/
drwxrwxr-x 2 ubuntu ubuntu 4096 Dec 1 20:46 lib/
-rw-r--r-- 1 ubuntu ubuntu 88906712 Oct 19 16:59 phoenix-4.8.1-HBase-1.1-client.jar
-rw-r--r-- 1 ubuntu ubuntu 62567971 Oct 19 16:58 phoenix-4.8.1-HBase-1.1-hive.jar
-rw-r--r-- 1 ubuntu ubuntu 6620780 Oct 19 16:58 phoenix-4.8.1-HBase-1.1-queryserver.jar
-rw-r--r-- 1 ubuntu ubuntu 24749543 Oct 19 16:59 phoenix-4.8.1-HBase-1.1-server.jar
-rw-r--r-- 1 ubuntu ubuntu 27070375 Oct 19 16:58 phoenix-4.8.1-HBase-1.1-thin-client.jar
ubuntu@VM-203-25-ubuntu:~/test/phoenix-4.8.1-HBase-1.1$
```

然后进入bin目录下，在配置文件hbase-site.xml中添加如下配置

```
<property>
<name>hbase.zookeeper.quorum</name>
<value>(腾讯云提供的连接地址和端口，管理控制台可查)</value>
</property>
<property>
<name>chbase.tencent.enable</name>
<value>true</value>
</property>
```

2.测试phoenix的可用性

在bin目录下可以使用shell来测试phoenix的可用性，./sqlline.py

```
ubuntu@VM-203-25-ubuntu:~/test/phoenix-4.8.1-HBase-1.1/bin$ ./sqlline.py
Setting property: [hbase.zookeeper.quorum, false]
Setting property: [isolation, TRANSACTION_READ_COMMITTED]
Issuing: [connect jdbc:phoenix: none none org.apache.phoenix.jdbc.PhoenixDriver
Connecting to jdbc:phoenix:
16/12/02 10:06:26 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Connected to: Phoenix (version 4.8)
Driver: PhoenixEmbeddedDriver (version 4.8)
Autocommit status: true
Transaction isolation: TRANSACTION_READ_COMMITTED
Building list of tables and columns for tab-completion (set fastconnect to true to skip)...
86/86 (100%) done
Done
sqlline version 1.1.9
0: jdbc:phoenix>
```

再输入!tables

```
0: jdbc:phoenix> !tables
+-----+-----+-----+-----+-----+-----+-----+-----+
| TABLE_CAT | TABLE_SCHEM | TABLE_NAME | TABLE_TYPE | REMARKS | TYPE_NAME | SELF_REFERENCING_COL_NAME | REF_GENERATION | INDEX_STATE |
+-----+-----+-----+-----+-----+-----+-----+-----+
| SYSTEM | SYSTEM | CATALOG | SYSTEM TABLE |  |  |  |  |  |
| SYSTEM | SYSTEM | FUNCTION | SYSTEM TABLE |  |  |  |  |  |
| SYSTEM | SYSTEM | SEQUENCE | SYSTEM TABLE |  |  |  |  |  |
| SYSTEM | SYSTEM | STATS | SYSTEM TABLE |  |  |  |  |  |
+-----+-----+-----+-----+-----+-----+-----+-----+
0: jdbc:phoenix>
```

3.基本SQL语句的使用

```

O: jdbc:phoenix:10.66.181.136:2181> CREATE TABLE stats.prod_metrics ( host char(50) not null created_date date not null
O: jdbc:phoenix:10.66.181.136:2181> UPSERT INTO stats.prod_metrics (host,created_date,txn_count) VALUES('192.168.0.1',TO_DATE('2016-12-02 10:00:00'),1);
1 row affected (0.284 seconds)
O: jdbc:phoenix:10.66.181.136:2181> SELECT * FROM stats.prod_metrics;
+-----+-----+-----+
| HOST      | CREATED_DATE | TXN_COUNT |
+-----+-----+-----+
| 192.168.0.1 | 2016-12-02 10:00:00.000 | 1 |
+-----+-----+-----+
O: jdbc:phoenix:10.66.181.136:2181> ALTER TABLE stats.prod_metrics ADD tip varchar(200);
No rows affected (6.003 seconds)
O: jdbc:phoenix:10.66.181.136:2181> UPSERT INTO stats.prod_metrics VALUES('192.168.1.1:2100',TO_DATE('2016-10-10 12:00:00'),1,'SOME OK');
1 row affected (0.031 seconds)
O: jdbc:phoenix:10.66.181.136:2181> SELECT * FROM stats.prod_metrics;
+-----+-----+-----+-----+
| HOST      | CREATED_DATE | TXN_COUNT | TIP |
+-----+-----+-----+-----+
| 192.168.0.1 | 2016-12-02 10:00:00.000 | 1 |  |
+-----+-----+-----+-----+
O: jdbc:phoenix:10.66.181.136:2181> UPSERT INTO stats.prod_metrics VALUES('192.168.0.1',TO_DATE('2016-10-10 12:00:00'),1,'SOME OK');
1 row affected (0.01 seconds)
O: jdbc:phoenix:10.66.181.136:2181> UPSERT INTO stats.prod_metrics VALUES('192.168.1.1:2100',TO_DATE('2016-10-10 12:00:00'),2,'SOME OK');
1 row affected (0.015 seconds)
O: jdbc:phoenix:10.66.181.136:2181> SELECT * FROM stats.prod_metrics;
+-----+-----+-----+-----+
| HOST      | CREATED_DATE | TXN_COUNT | TIP |
+-----+-----+-----+-----+
| 192.168.0.1 | 2016-10-10 12:00:00.000 | 1 | SOME OK |
| 192.168.0.1 | 2016-12-02 10:00:00.000 | 1 |  |
| 192.168.1.1:2100 | 2016-10-10 12:00:00.000 | 2 | SOME OK |
+-----+-----+-----+-----+

```

<http://phoenix.apache.org/language/index.html>

各语言支持

支持多语言访问

若需要通过多语言的方式访问HBase，需要开启shtrift 服务。通过如下方式，进入HBase目录（连接和访问里有下载地址，并需要配置云参数），然后通过如下方式启动

```
bin/hbase-daemon.sh start thrift -p <port> --infoport <infoport>
```