

**中国电信大数据平台SparkSql查询支持**

**实现说明书**

**中国电信集团公司**

**2016年5月**

**修订记录**

|  |  |  |  |
| --- | --- | --- | --- |
| 修订时间 | 编撰者 | 版本 | 修订内容 |
| 2016-05-12 | 李刚 | V0.5 | 创建 |
|  |  |  |  |

**目录**

[1 SparkSql 5](#_Toc31389)

[1.1 概况 5](#_Toc1063)

[1.2 案例 7](#_Toc28656)

[2 Livy server实现SparkSql 8](#_Toc28904)

[2.1 Livy server的架构原理 8](#_Toc13705)

[2.2 Livy server实现SparkSql的方案 11](#_Toc19904)

[2.3 Livy server实现SparkSql的优缺点 11](#_Toc23741)

[3 Web rest服务实现SparkSql 12](#_Toc19212)

[3.1 Web rest服务实现SparkSql方案 12](#_Toc1977)

[3.2 Web rest服务实现SparkSql要解决的问题 13](#_Toc28311)

[4 beeswax来实现SparkSql 14](#_Toc2110)

[4.1 beeswax介绍 14](#_Toc8627)

[4.2 beeswax实现SparkSql 14](#_Toc28298)

[4.3 beeswax实现SparkSql的缺点 14](#_Toc5903)

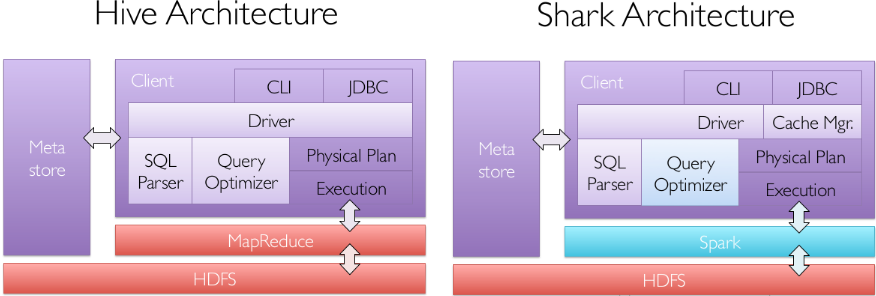
# SparkSql

## 概况

SparkSQL的前身是shark。在hadoop发展过程中，为了给熟悉RDBMS但又不理解MapReduce的技术人员提供快速上手的工具，hive应运而生，是当时唯一运行在hadoop上的SQL-on-Hadoop工具。但是，MapReduce计算过程中大量的中间磁盘落地过程消耗了大量的I/O，降低的运行效率，为了提高SQL-on-Hadoop的效率，大量的SQL-on-Hadoop工具开始产生，其中表现较为突出的是：

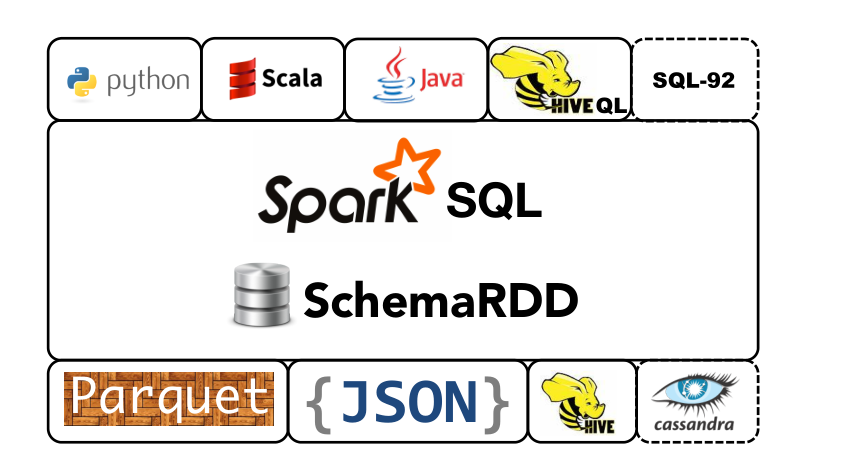
1. MapR的Drill
2. Cloudera的Impala
3. Shark

其中Shark是伯克利实验室spark生态环境的组件之一，它修改了下图所示的右下角的内存管理、物理计划、执行三个模块，并使之能运行在spark引擎上，从而使得SQL查询的速度得到10-100倍的提升。



但是，随着Spark的发展，对于野心勃勃的Spark团队来说，Shark对于hive的太多依赖（如采用hive的语法解析器、查询优化器等等），制约了Spark的One Stack rule them all的既定方针，制约了spark各个组件的相互集成，所以提出了sparkSQL项目。SparkSQL抛弃原有Shark的代码，汲取了Shark的一些优点，如内存列存储（In-Memory Columnar Storage）、Hive兼容性等，重新开发了SparkSQL代码；由于摆脱了对hive的依赖性，SparkSQL无论在数据兼容、性能优化、组件扩展方面都得到了极大的方便。

SparkSql的架构图如下：



## Spark SQL组件

使用Spark SQL时，最主要的两个组件就是DataFrame和SQLContext。

首先，我们来了解一下DataFrame。

### DataFrame：[DataFrame](https://spark.apache.org/docs/1.3.0/api/scala/index.html" \l "org.apache.spark.sql.DataFrame)是一个分布式的，按照命名列的形式组织的数据集合。DataFrame基于R语言中的data frame概念，与关系型数据库中的数据库表类似。之前版本的Spark SQL API中的SchemaRDD已经更名为DataFrame。

通过调用将DataFrame的内容作为行RDD（RDD of Rows）返回的[rdd方法](https://spark.apache.org/docs/1.3.0/api/scala/index.html" \l "org.apache.spark.sql.DataFrame)，可以将DataFrame转换成RDD。

可以通过如下数据源创建DataFrame：

1. 已有的RDD
2. 结构化数据文件
3. JSON数据集
4. Hive表
5. 外部数据库

本篇文章仅仅是针对Hive表中的数据使用SparkSql来查询。

Spark SQL和DataFrame API已经在下述几种程序设计语言中实现：

* Scala
* Java
* Python

### SQLContext

Spark SQL提供[SQLContext](http://spark.apache.org/docs/latest/api/scala/index.html" \l "org.apache.spark.sql.SQLContext)封装Spark中的所有关系型功能。可以用之前的示例中的现有SparkContext创建SQLContext。下述代码片段展示了如何创建一个SQLContext对象。

val sqlContext = new org.apache.spark.sql.SQLContext(sc)

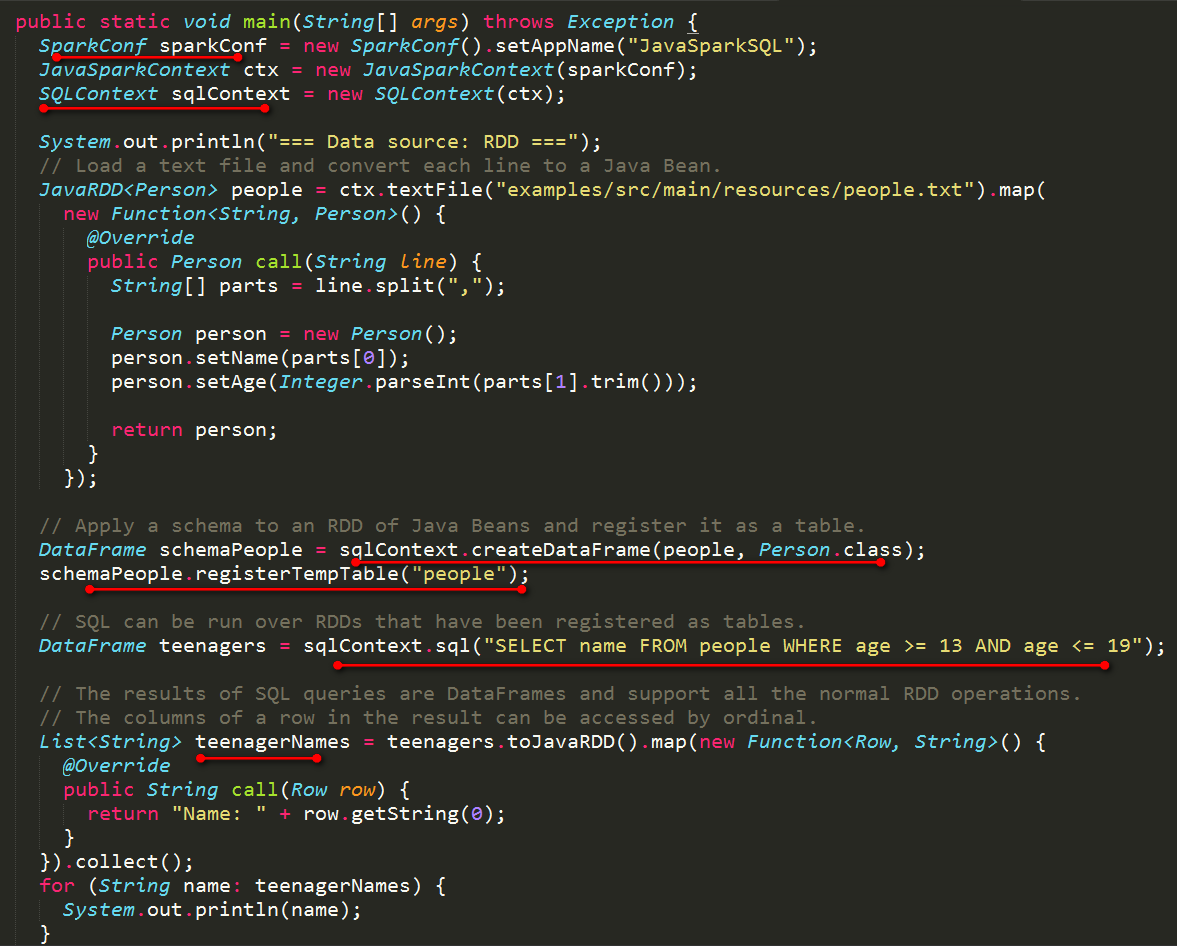
此外，Spark SQL中的[HiveContext](https://spark.apache.org/docs/1.3.0/api/scala/index.html" \l "org.apache.spark.sql.hive.HiveContext)可以提供SQLContext所提供功能的超集。可以在用HiveQL解析器编写查询语句以及从Hive表中读取数据时使用。

在Spark程序中使用HiveContext无需既有的Hive环境。

## 案例

下面通过2个案例来说明如何访问SparkSql服务。

1. 案例1：通过java代码来执行相关SparkSql服务



重点就是通过构建出SQLContext来执行它的sql方法即可。数据源可以是上面提到的那几种数据源，这里并不限制。

代码的执行是通过spark的spark-submit命令来执行该main函数的。

1. 案例2：通过JDBC方式来访问和执行相关SparkSql服务

如果数据源是Hive，则SparkSql还支持通过JDBC方式来连接SparkSql来进行Hive数据的查询。步骤如下：

1. 启动SparkSql的thrift服务

sbin/start-thriftserver.sh

该服务其实就是在Hive的HiveServer2之上包装了一层，用于支持JDBC方式来连接SparkSql服务进行查询。

1. 通过beeline客户端就可以来连接上述thrift服务了

bin/beeline

beeline> !connect jdbc:hive2://localhost:10000

这是通过命令行的方式来连接，如果通过java代码的方式，就要通过如下方式来实现JDBC连接：

Driver:org.apache.hive.jdbc.HiveDriver

Url:jdbc:hive2://localhost:10000

user:运行用户

Password:随意

这种JDBC方式的底层实现原理是否是通过SQLContext的sql方法来实现的呢？这个还需要进一步验证确认。

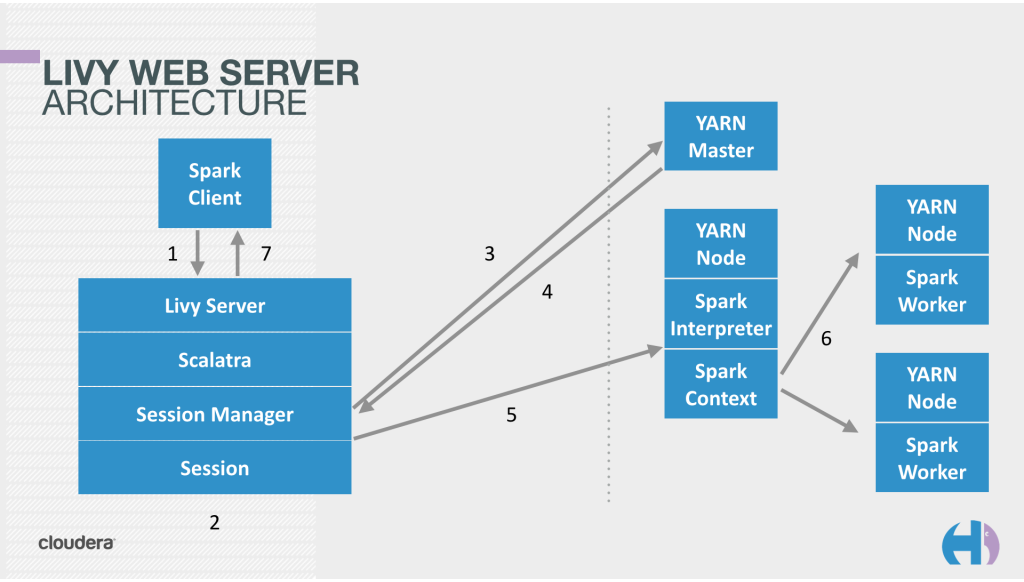
# Livy server实现SparkSql

## Livy server的架构原理

Livy server是针对Spark的开源的REST接口，使得我们可以通过REST接口来实现与Spark交互。具有如下功能：

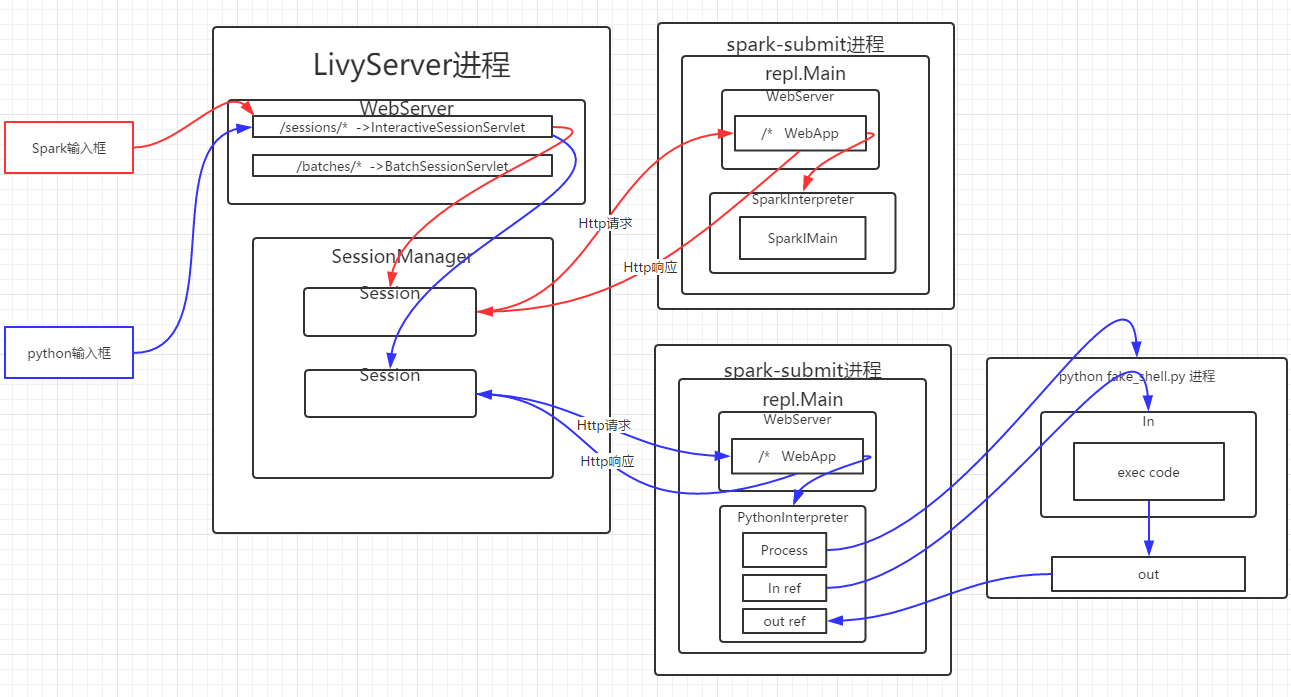
1. 可以与scala、python、R shell客户端交互，执行一些代码片段
2. 可以提交整个Spark Job,支持scala、python、java编写的Spark job。

来看下Livy server的整体架构图：



Livy Server内部启动了一个web服务，使用的是scala中的Scalatra的web框架。内部还拥有一个SessionManager来管理着与用户之间保持的session。

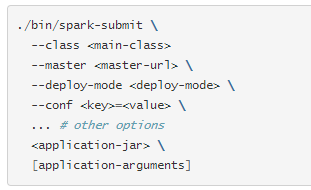
Session会将用户要执行的代码通过spark-submit来提交给yarn，然后执行。这仅仅是整体的概念，再来看下详细的Livy server内部组件图：



上述的Spark输入框和python输入框分别代表用户在hue界面上开启一个相关的session会话，即livy server都会为他们创建出一个session，不同类型的session启动不同的服务。下面分别以spark、python来详细说明。

Spark输入框：

1. 创建出一个kind为scala类型的Session（scala、spark都属于该类型）
2. Session创建后，会通过spark-submit命令启动一个新的进程。这里来详细看下spark-submit命令的先关参数：



上述--class内容在这里就是 com.cloudera.hue.livy.repl.Main

启动参数就是Session的类型，如scala、python等

1. com.cloudera.hue.livy.repl.Main启动后，内部又启动了一个web服务，用来处理http请求，该web服务只有一个Servlet就是WebApp，该Servlet用于执行代码片段、查询代码片段执行结果等等。
2. WebApp在执行代码片段的时候，依托于内部的的解释器Interpreter。上述com.cloudera.hue.livy.repl.Main的启动参数中指明了类型，不同的类型就创建出对应的解释器，分别有如下三种解释器：

SparkInterpreter

PythonInterpreter

SparkRInterpreter

这里我们就重点关注下SparkInterpreter，SparkInterpreter内部是通过Spark的SparkIMain来编译、执行、绑定数据的。

1. 一旦SparkSubmit进程启动完毕，就会回调Livy server告诉对应的Session已准备完毕，并且告知该Session SparkSubmit进程启动的web服务地址，供session发送代码片段时使用。对应Session就设置自己的状态为idle空闲状态，此时用户在hue界面上就显示会话创建完毕。
2. 用户输入相关代码点击执行，则通过Http请求发送给对应的session来处理，session创建出statement语句来表示该代码片段，并返回statement语句id作为查询的标示。Session同时将该代码片段通过Http请求发送给对应的SparkSubmit进程的web服务
3. SparkSubmit进程的WebApp接收到该Http请求，从中取出代码片段，交给SparkInterpreter内部的SparkIMain来编译、执行和绑定相关数据。
4. 用户发送代码片段后，hue就在不断的查询代码片段的执行结果，一旦执行完成，就在hue上显示出来。

这种代码片段是可以有上下文信息的，得益于SparkIMain的绑定，如代码片段中可以直接使用SparkContext，因为SparkIMain在启动的时候就创建出了SparkContext并绑定到上下文环境中了。

Python输入框：

大部分流程和上述一样的。但是启动的新的SparkSubmit进程在初始化PythonInterpreter的时候又启动了一个新的进程python fake\_shell.py 。

并且获取了新进程的输入和输出。

该PythonInterpreter执行代码的过程就是向新python进程的输入要执行的代码，新的python进程就在不断的while循环中获取输入的代码信息并使用python中的exec来动态执行代码片段。

这两种方式的区别：第一种方式算是使用SparkIMain类库的方式来交互执行代码，而python这种就是通过新进程的方式来交互执行。那么这时候对于SparkSql我们也可以采用类似python的这种方式来实现。

## Livy server实现SparkSql的方案

就是可以借鉴python那种启动新的专门进程的方式来实现交互。可如下实现步骤：

1. 实现一个SparkSqlInterpreter,该SparkSqlInterpreter在初始化的时候启动spark-sql这个进程。spark-sql进程是spark提供的用于执行sql的命令行客户端。它的实现原理是：

spark-submit

--class org.apache.spark.sql.hive.thriftserver.SparkSQLCLIDriver

SparkSQLCLIDriver内部创建了HiveContext，使用HiveContext来执行相关sql语句，这个执行sql的原理和上述介绍的案例1是类似的。它不需要thrift服务，仅仅需要hive的metastore存储服务。

1. spark-sql进程是交互式的，执行用户输入的sql语句，并且含有上下文信息。这个就跟python fake\_shell.py进程的作用是一致的
2. SparkSqlInterpreter掌管spark-sql进程的输入和输出，给spark-sql进程输入sql命令，spark-sql进程输出相关结果。这样的话，就完全吻合livy server本身的设计流程，不需要前端再更改任何逻辑，只需要多添加一个spark-sql类型即可。

## Livy server实现SparkSql的优缺点

优点：

1. 实现一个SparkSqlInterpreter后，就完全吻合livy server自身的设计流程，不需要前端再改动任何逻辑（前端逻辑有创建会话、检查会话状态、创建语句、检查语句执行结果等等一些逻辑），只需要多添加一个选项即可。

缺点：

1. livy server实现一个高质量的考虑全面的SparkSqlInterpreter有一定的开发难度。
2. spark-sql的输出结果目前是图形化打印出来的，不太好获取到结果数据，这个就需要修改原有输出方式，将数据以json字符串形式输出。
3. Livy server目前的实现方式：用户每打开一个输入框，就创建一个会话，并且又启动了一个SparkSubmit进程，如果是python的话还会启动python fake\_shell.py进程，这样的好处就是为每个用户保存了上下文信息，但是这样的处理方式在用户量过大的时候，应该是撑不住的。
4. Livy server目前已经完全从hue里面独立出来了，成为了一个cloudera的独立项目，新项目目前实现方式和上述hue中的livy server差别较大，如上述SparkSubmit进程中不再是web服务形式，而是使用netty来实现RPC服务。
5. livy server对session的管理是：1小时内未被使用就关闭掉相关进程服务。但是Hue中livy server还没有注册shutdownhook，以至于livy server关闭时，未能正常关闭其启动的相关进程服务，导致一直残存，占用资源。

# Web rest服务实现SparkSql

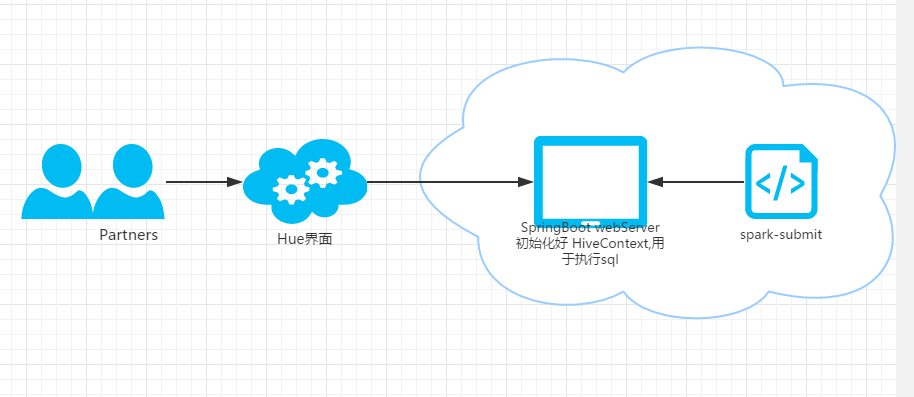
## Web rest服务实现SparkSql方案

这个其实底层使用SQLContext来执行sql查询，针对Hive数据源，则使用HiveContext。

spark-submit命令执行的代码完毕就结束了，所以我们需要开启一个web服务供外部来调用，使得spark-submit命令不会终止，变成一个长久服务。Main函数里面启动一个web服务，跟一般的Tomcat web服务就不太一样，就需要使用嵌入式的web服务器，我们选择使用SpringBoot内嵌tomcat服务器来开发实现。

使用spark-submit脚本启动一个main class，启动过程中我们创建出HiveContext对象，然后使用内嵌的web服务来接收http形式的sql请求，使用HiveContext来执行该sql即可。

简单的访问方式如下图所示：



这种方式就和livy server没啥关系了，不走livy server的流程了，但是需要一个自己的界面。这仅仅是一个demo功能，还有很多问题要解决。

## Web rest服务实现SparkSql要解决的问题

1. 上述形式仅仅是执行一个sql语句，就没有Livy server中会话保持功能了，即不再会有上下文信息。

如果要想实现会话保持功能，就需要跟livy server的做法一样，用户新打开一个输入框，就需要创建出一个会话，就创建出一个HiveContext。目前spark中只允许有一个SparkContext对象，多个会有一定的bug。但是我们是可以使用同一个SparkContext来创建多个HiveContext的，还需要进一步查看源代码确认多个HiveContext是否有共享数据问题。

1. HiveContext的创建过程还是很耗时的，所以会话的创建就需要是异步创建，直接返回给用户会话ID,用户拿到会话ID需要不断检查该会话是否已创建完毕。这个和livy server中的session的创建是类似的。
2. 会话的管理：简单策略就是超过一定时间没有再使用，就关闭该会话。这个和livy server中SessionManager的管理是类似的。
3. HiveContext可以通过use db这个sql命令来切换HiveContext使用的数据库，这个可能需要和hive查询界面保持一致，不允许这样方式来切换数据库。则需要禁止掉上述sql命令。
4. Sql语句的执行也是很耗时的，需要异步执行，这样的话就需要创建类似livy server中的Statement的对象用来保存sql语句和对应的数据库，异步执行sql语句，直返返回给用户相关ID，界面中就不断轮训去查询语句执行情况，执行完毕就可以查到相关结果并展示。这一部分和livy server中Session中的Statement的维护是类似的。
5. HiveContext的缓存问题：有时候查询使用了缓存，查询结果很快就出来，但是有时候没用到缓存，耗时比较长，还需要深入研究下HiveContext的缓存机制。

可以看到如果是自己实现web rest服务，还是要做很多和livy server类似的东西的，并且需要一个对应的前端界面，来实现查询session是否建立成功、sql语句执行情况等配套逻辑。

# beeswax来实现SparkSql

## beeswax介绍

beeswax原本就是用来配置hive查询的，原理就是连接hive的thrift服务来执行hql查询。

## beeswax实现SparkSql

Spark也实现了一个thrift服务，原理就是在hive上述thrift服务基础上包装了一层，并创建spark的HiveContext对象，使用该HiveContext对象来执行相关的sql查询。

所以只需要将beeswax所连接的thrift服务更改成spark的thrift服务即可实现SparkSql查询。这种方式不用更改任何代码，但是前提是：要使用hue3.9版本。

## beeswax实现SparkSql的缺点

1. 直接修改beeswax的thrift地址，就相当于废弃了hive的thrift服务，还需要研究下是否beeswax是否支持多种thrift服务共存的情况。代码中貌似是支持的，这个还需要继续确认下。
2. 在hue3.9的notebook界面中进行SparkSql查询的原理是：每次执行sql语句之前，都要先执行下use db来切换相应的数据库，然后再执行sql语句。这种做法是由于Spark实现的thrift服务中只创建了一个HiveContext，这个HiveContext对象是大家共享的，所以就会造成不同用户之间相互干扰的情况。这个是非常致命的问题。