

# 一.UDF基础知识

## 1.1基础

**HIVE总结**

**HIVE的用途、实现原理、架构、安装、用法？**

公司的数据处理方式（四种主流）：MR、HIVE、HBase、Spark....

非主流：pig、storm、mongodb、mr script.....

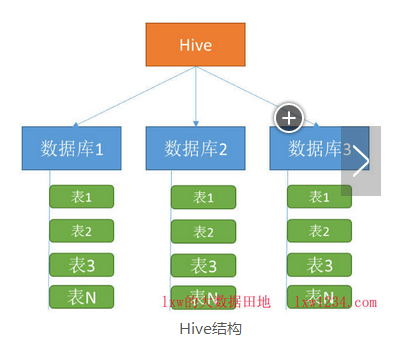
流程：

1.使用MR开发：编写MR ，实现mapper、reducer、main在hadoop上运行

2.使用hive开发：

2.1使用内置函数：在hive的CLI模式下写HQL，自动转成MR，在hadoop上运行

2.2使用UDF自定义函数：写class打jar包，在hive的CLI下添加自定义函数，在CLI模式下写HQL，自动转成MR，在hadoop上运行



**hive知识点总结：**

Hive是建立在Hadoop上的数据仓库基础构架（HDFS和MR），它提供了一系列工具可以用来进行数据提取、转化、加载，这是一种可以存储、查询和分析存储在Hadoop中的大规模的数据机制。

Hive就是在Hadoop上架了一层SQL接口，可以将SQL翻译成MapReduce去Hadoop上执行，这样就使得数据开发和分析人员很方便的使用SQL来完成海量数据的统计和分析，而不必使用编程语言开发MapReduce那么麻烦

Hive是基于HDFS存储、基于MapReduce计算，提供HQL查询的数据仓库工具。Hive是一个很开放的系统，很多内容都支持用户定制。

HQL：使用语言：QL语言（类SQL查询语言）。能处理内建的mapper和reducer无法完成的复杂分析工作。

现在虽然有很多SQL ON Hadoop的解决方案，像Spark SQL、Impala、Presto等等，但就目前来看，在基于Hadoop的大数据分析平台、数据仓库中，Hive仍然是不可替代的角色。尽管它的相应延迟大，尽管它启动MapReduce的时间相当长，但是它太方便、功能太强大了，做离线批量计算、ad-hoc查询甚至是实现数据挖掘算法，而且，和HBase、Spark都能整合使用。

1.Hive是SQL解析引擎，它将SQL语句转译成M/R Job然后再Hadoop执行。

2.Hive的表其实是HDFS的目录（/文件夹），按表名把文件夹区分开。如果是分区表，则分区值是子文件夹，可以直接在M/R Job里使用这些数据。Hive的数据都是存储在HDFS上的，默认有一个根目录，在hive-site.xml中，由参数hive.metastore.warehouse.dir指定。默认值为/user/hive/warehouse.

例如：创建logs数据库，HDFS上的存储地址就是/user/hive/warehouse/logs.db

例如：创建day表，HDFS上的存储地址就是/user/hive/warehouse/logs.db/day/

**Hive的系统架构**

1.用户接口主要有三个：CLI，JdbC/ODBC，WebUI

（1）CLI，即shell命令行。

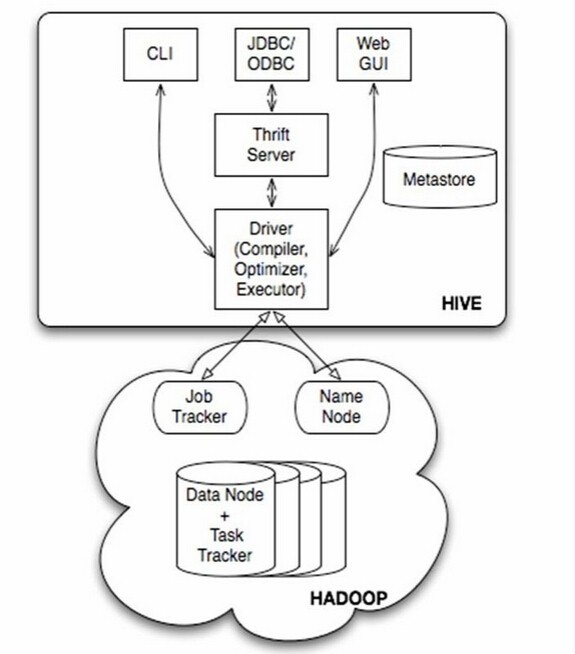
（2）JDBC/ODBC，是Hive的Java接口，与使用传统数据库JDBC方式类似。

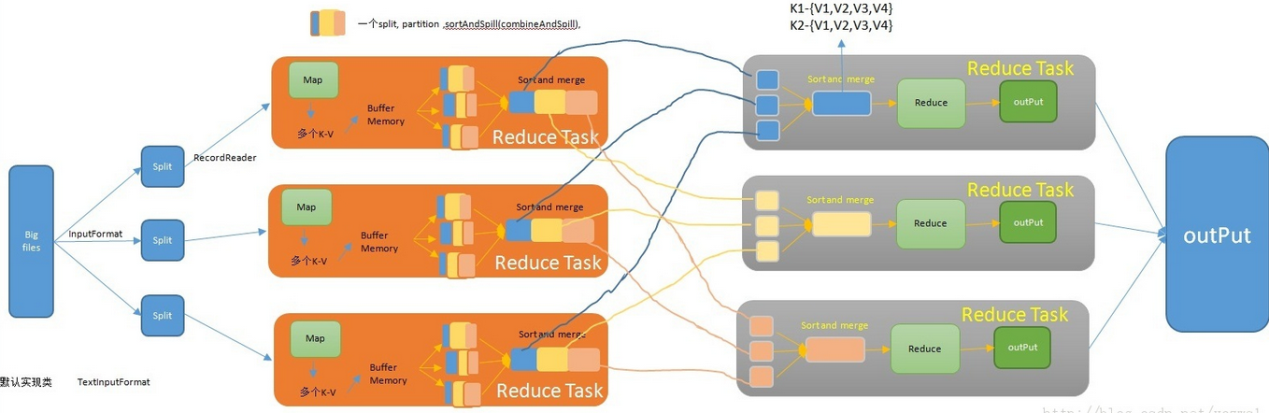
（3）WebUI，是通过浏览访问Hive。

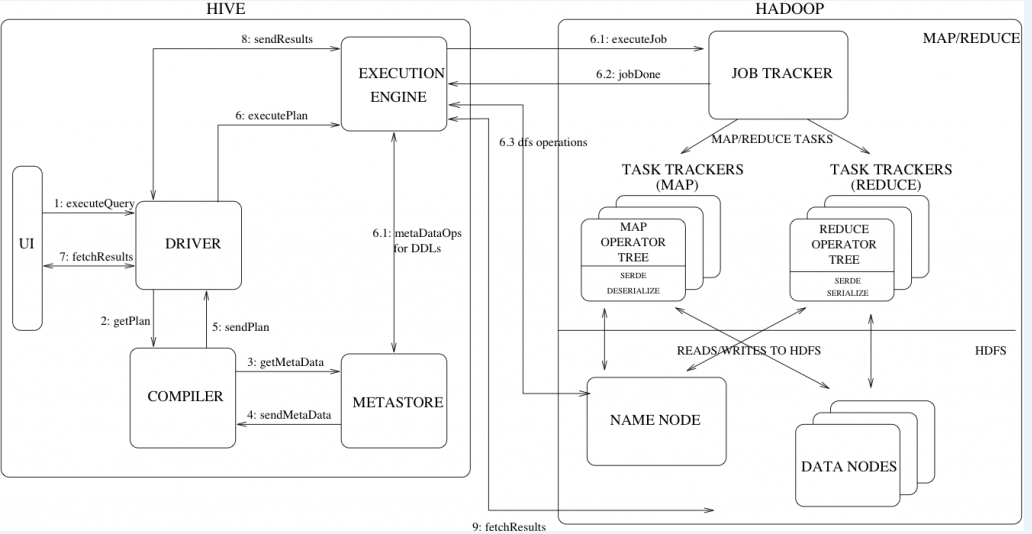
2.Hive将元数据存储在数据库（metastore）中，目前只支持MySQL，derby。（Hive中的元数据包括表名，表列，和分区及其属性，表属性，表的数据所在目录等）

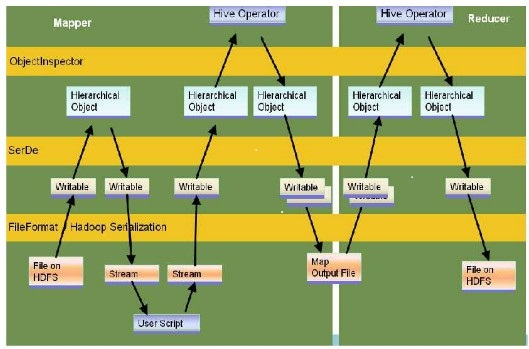
3.解析器、编译器、优化器、执行器完成HQL查询语句从词法分析、语法分析、编译、优化以及查询计划（plan）的生成。生成的查询计划存储在HDFS中，并在随后有MapReducer调用执行。

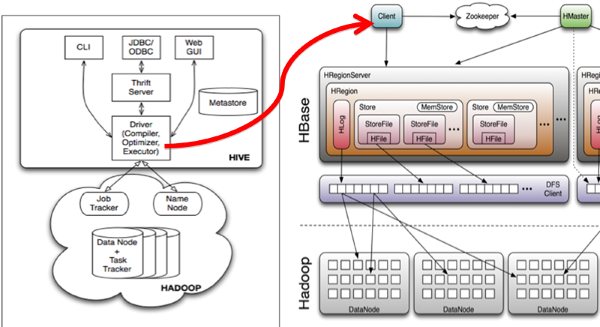
4.Hive的数据存储在HDFS中，大部分的查询由MapReducer完成（包括含\*的查询，比如select \* from table不会生成MapReducer任务）











Hive通过给用户提供的一系列交互接口，接收到用户的指令(SQL)，使用自己的Driver，结合元数据(MetaStore)，将这些指令翻译成MapReduce，提交到Hadoop中执行，最后，将执行返回的结果输出到用户交互接口。

在使用过程中，至需要将Hive看做是一个数据库就行，本身Hive也具备了数据库的很多特性和功能。

**hive的数据单元**

Databases：数据库。概念等同于关系型数据库的Schema，不多解释；

Tables：表。概念等同于关系型数据库的表，不多解释；

Partitions：分区。概念类似于关系型数据库的表分区，没有那么多分区类型，只支持固定分区，将同一组数据存放至一个固定的分区中。

Buckets (or Clusters):分桶。同一个分区内的数据还可以细分，将相同的KEY再划分至一个桶中，这个有点类似于HASH分区，只不过这里是HASH分桶，也有点类似子分区吧。

**hive的数据类型**

基本类型

复杂类型

**hive与sql数据库的区别**

数据库的操作是CRUD 增删改查

hive的操作是create table,upload data,select col,drop table。CRUD

事物问题

**hive与mr的比较**

1. 运算资源消耗

无论从时间，数据量，计算量上来看，一般情况下mr都是优于或者等于hive的。mr的灵活性是毋庸置疑的。在转换到hive的过程中，会有一些为了实现某些场景的需求而不得不用多步hive来实现的时候。

2. 开发成本/维护成本

hive的开发成本是远低于mr的。

3. 底层相关性

在使用hive以后，读取文件的时候，再也不用关心文件的格式，文件的分隔符，只要指定一次，hive就会保存好。相比mr来说方便了很多。

**hive的优缺点**

1.hive独有的分区管理，方便进行数据的管理 优

2.代码的管理也很方便，就是直接的文本 优

3.逻辑的修改和生效很方便 优

4.但是当出现异常错误的时候，hive的调试会比较麻烦。特别是在大的生产集群上面的时候 缺

5. 当侧重关心与业务相关的内容的时候，用hive会比较有优势。而在一些性能要求高，算法研究的时候，mr会更加适合 缺

6. 批处理，高延迟的计算框架 缺

7. 内置的函数和自定义函数开发 优

8. Hive对事物的支持不够好，原因是HDFS本身就设计为一次写入，多次读取的分布式存储系统，因此，不能使用Hive来完成诸如DELETE、UPDATE等在线事务处理的需求

9. Hive擅长的是非实时的、离线的、对响应及时性要求不高的海量数据批量计算，即查询，统计分析。

**总结：实际上HIVE基于Hadoop（HDFS+MR）和SQL数据库的一个工具**

**UDF自定义函数**

三种自定义函数：UDF、UDAF、UDTF

UDF(user-defined function):操作单行数据，并产生一个数据行作为输出。大多数函数，常用（数值函数、字符串函数）

UDAF(user-defined aggregate funtion):接受多个输入数据行，并产生一个输出数据行 不常用（聚合函数）

UDTF(user-defined table-generation):操作单个数据行，并且产生多个数据行（一个表）不用

## 1.2Hive UDF开发

使用自定义函数的目的就是更好的处理数据。

Hive进行UDF开发十分简单,Hive自身查询语言HQL能完成大部分的功能，但遇到特殊需求时，需要自己写UDF实现。所以需要hive版本在0.4.0以上才可以。

开发自定义UDF函数有两种方式，一个是继承org.apache.hadoop.hive.ql.exec.UDF，另一个是继承org.apache.hadoop.hive.ql.udf.generic.GenericUDF；

如果是针对简单的数据类型（比如String、Integer等）可以使用UDF，如果是针对复杂的数据类型（比如Array、Map、Struct等），可以使用GenericUDF，另外，GenericUDF还可以在函数开始之前和结束之后做一些初始化和关闭的处理操作。

编写Hive UDF有两种方式：

1. extends UDF ， 重写evaluate方法 常用

2. extends GenericUDF，重写initialize、getDisplayString、evaluate方法 不用

用户可以使用‘show functions’ 查看function list，可以使用'describe function function-name'查看函数说明。

e.g

package com.hrj.hive.udf;

import org.apache.hadoop.hive.ql.exec.UDF;

public class helloUDF extends UDF {

public String evaluate(String str) {

try {

return "HelloWorld " + str;

} catch (Exception e) {

return null;

}

}

}

## 1.3自定义函数调用过程：

将该java文件编译成helloudf.jar

①添加jar包（在hive命令行里面执行）

hive> add jar helloudf.jar;

②创建临时函数 ，hive命令行关闭后，即失效。

hive> create temporary function helloworld as 'com.hrj.hive.udf.helloUDF';

③调用

hive> select helloworld(t.col1) from t limit 10;

④将查询结果保存到HDFS中

hive> create table result row format delimited fields terminated by '\t' as select id, testlength(nation) from student;

5.将函数注销

hive> drop temporary function helloworld;

注：

1.helloworld为临时的函数，所以每次进入hive都需要add jar以及create temporary操作

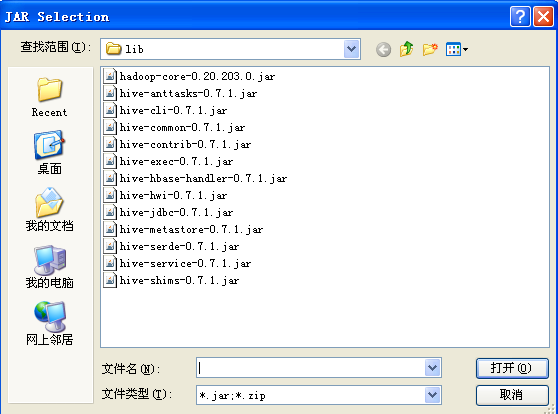
2.UDF只能实现一进一出的操作，如果需要实现多进一出，则需要实现UDAF

## 1.4 开的UDF用的是jar包

常用：

hive-exec-1.2.0.jar 基础包

hadoop-common-2.7.1.jar 类型包



## 1.5 UDF开发步骤：

1. 新建java项目，导入从安装好的hive目录下拷贝hive-exec-xxx.jar包并导入项目（其他包按需导入即可）

2. 新建普通Java类，并继承UDF类

3. 在类中实现evaluate方法

注意事项

1. evaluate方法可以有多个，进行方法的重载

2. evaluate方法的返回值类型不能为void，但是返回值可以为null

3. evaluate方法的参数可有可无

hiveUDF使用方法

1.临时注册（每次重新打开hive都需要注册）

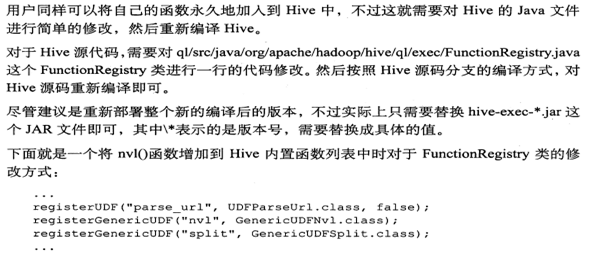
1.1.将写好的java项目导出成jar包，导出的时候不需要连lib包一起导出，因为这些jar包 hive里都有

1.2.将jar包放置在一个进群中某台可运行hive的机器的本地目录

1.3.注册jar包（将jar包添加到hive中）

1.4.给自己所写的方法起个名字

2.永久使用（将其添加到hive中去，需要重新编译hive）



路径：\ql\src\java\org\apache\hadoop\hive\ql\exec\FunctionRegistry.java

添加内容：registerUDF("hash\_code", UDFHashCode.class, false);

重新编译hive，很麻烦 不用这个方法。

和前两者相比，第三种方式直接将用户的自定义函数作为注册为内置函数，未来使用起来非常简单，但这种方式也非常危险，一旦出错，将是灾难性的，因此，建议如果不是特别通用，并且固化下来的函数，还是使用前两种方式比较靠谱。

## 1.6 Hive UDAF开发

Hive进行UDAF开发，相对要比UDF复杂一些。Hive查询数据时，有些聚类函数在HQL没有自带，需要用户自定义实现。

## 1.7 开发UDAF

两个包是必须的import org.apache.hadoop.hive.ql.exec.UDAF和 org.apache.hadoop.hive.ql.exec.UDAFEvaluator

请看一个例子

package org.hrj.hive.udf;

import org.apache.hadoop.hive.ql.exec.UDAFEvaluator;

import org.apache.hadoop.hive.serde2.io.DoubleWritable;

public class UDAFSum\_Sample extends NumericUDAF {

public static class Evaluator implements UDAFEvaluator {

private boolean mEmpty;

private double mSum;

public Evaluator() {

super();

init();

}

public void init() {

mSum = 0;

mEmpty = true;

}

public boolean iterate(DoubleWritable o) {

if (o != null) {

mSum += o.get();

mEmpty = false;

}

return true;

}

public DoubleWritable terminatePartial() {

// This is SQL standard - sum of zero items should be null.

return mEmpty ? null : new DoubleWritable(mSum);

}

public boolean merge(DoubleWritable o) {

if (o != null) {

mSum += o.get();

mEmpty = false;

}

return true;

}

public DoubleWritable terminate() {

// This is SQL standard - sum of zero items should be null.

return mEmpty ? null : new DoubleWritable(mSum);

}

}

}

1.将java文件编译成Sum\_Sample.jar

2.进入hive

hive> add jar Sum\_sample.jar;

hive> create temporary function sum\_test as 'com.hrj.hive.udf.UDAFSum\_Sample';

hive> select sum\_test(t.num) from t;

hive> drop temporary function sum\_test;

hive> quit;

关于UDAF开发注意点：

1.需要import org.apache.hadoop.hive.ql.exec.UDAF以及org.apache.hadoop.hive.ql.exec.UDAFEvaluator,这两个包都是必须的

2.函数类需要继承UDAF类，内部类Evaluator实现UDAFEvaluator接口

3.Evaluator需要实现 init、iterate、terminatePartial、merge、terminate这几个函数

1）init函数类似于构造函数，用于UDAF的初始化

2）iterate接收传入的参数，并进行内部的轮转。其返回类型为boolean

3）terminatePartial无参数，其为iterate函数轮转结束后，返回乱转数据，iterate和terminatePartial类似于hadoop的Combiner

4）merge接收terminatePartial的返回结果，进行数据merge操作，其返回类型为boolean

5）terminate返回最终的聚集函数结果

# 2. 总结

1、重载evaluate函数。

2、UDF函数中参数类型可以为Writable，也可为java中的基本数据对象。

3、UDF支持变长的参数。

4、Hive支持隐式类型转换。

5、客户端退出时，创建的临时函数自动销毁。

6、evaluate函数必须要返回类型值，空的话返回null，不能为void类型。

7、UDF是基于单条记录的列进行的计算操作，而UDFA则是用户自定义的聚类函数，是基于表的所有记录进行的计算操作。

8、UDF和UDAF都可以重载。

9、查看函数

SHOW FUNCTIONS;

DESCRIBE FUNCTION <function\_name>;