# Hive基本概念

## 1 什么是Hive

**1） hive简介**

**Hive:**用于解决海量结构化日志的数据统计工具。

Hive是基于Hadoop的一个数据仓库工具，可以将结构化的数据文件映射为一张表，并提供类SQL查询功能。

**2） Hive本质**：将HQL转化成MapReduce程序

运行MapReduce程序，生成相应的分析结果,将结果以表的形式给到client

client

用户根据业务需求编写相应的SQL语句

Hive将SQL语言中常用的操作（select，where，group等）用MapReduce写成很多模板

通过Hive框架匹配出相应的MapReduce模板

所有的MapReduce模板封装在Hive中

（1）Hive处理的数据存储在HDFS

（2）Hive分析数据底层的实现是MapReduce

（3）执行程序运行在Yarn上

Hive的优点：

（1）操作接口采用类SQL语法，提供快速开发的能力（简单、容易上手）。

（2）避免了去写MapReduce，减少开发人员的学习成本。

（3）Hive常用于数据分析，对实时性要求不高的场合。

（4）Hive优势在于处理大数据，对于处理小数据没有优势，因为Hive的执行延迟比较高。

（5）Hive支持用户自定义函数，用户可以根据自己的需求来实现自己的函数。

Hive架构原理



（1）解析器：将SQL字符串转换成抽象语法树AST，解析器会对AST进行语法分析，比如表是否存在、字段是否存在、SQL语义是否有误。

（2）编译器：将AST编译生成逻辑执行计划。

（3）优化器：对逻辑执行计划进行优化。

（4）执行器：把逻辑执行计划转换成可以运行的物理计划，也就是断定当前MR的执行是否需要基于上一步MR的结果。对于Hive来说，就是MR/Spark。

**Metastore：hive元数据(表名，表的数据所在目录等)存储，hive中表的数据不在表中，而是在hdfs中，所以表会与hdfs中的数据产生映射关系。**

**默认存在自带的derby数据库中，但我们用mysql。**

Hive的运行机制：

HQL -> Hive ->数据库查元数据->确定某表与hdfs中某文件存在关联->hive分析数据

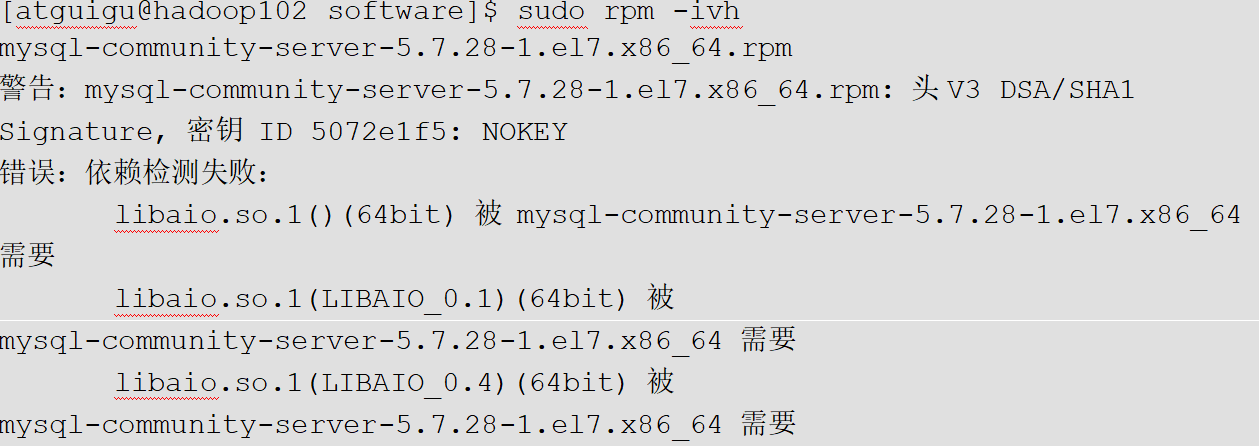
Hive不是数据库！！！只是操作起来比较像，Hive的定义是数仓分析工具！并且数据库不能处理海量数据。

# Hive安装

## 1.MySql安装

为啥需要mysql？原因在于Hive默认使用的元数据库为derby，开启Hive之后就会占用元数据库，且不与其他客户端共享数据，如果想多窗口操作就会报错，操作比较局限。以我们需要将Hive的元数据地址改为MySQL，可支持多窗口操作。

1. **检查当前系统是否安装过Mysql并卸载**
   * + 1. 查询：rpm -qa|grep mariadb（CentOS6->mysql CentOS7 ->mariadb）
       2. **卸载：rpm -qa | grep mariadb | xargs -n1 sudo rpm -e --nodeps**
2. **安装**
   * + 1. **将MySQL的tar包拷贝到/opt/software目录下**
       2. **解压MySQL安装包**
       3. **在安装目录下执行rpm安装（按顺序执行）**
          1. **common**
          2. **libs**
          3. **libs-compat**
          4. **client**
          5. **server(安装之前注意安装依赖)**



依赖下载：yum install -y libaio

* + - 1. **初始化数据库**

sudo mysqld --initialize --user=mysql

* + - 1. **启动MySQL服务**

sudo systemctl start mysqld

* + - 1. **登录MySQL数据库**

mysql -uroot -p密码

set password = password("新密码")

* + - 1. 解决MySql的登陆问题

mysql -uroot -p密码 -hhadoop101

更改只能本地链接问题

刷新权限：flush privileges;

* + - 1. 涉及中文编码：show variables like “character%”;

## 2.Hive安装

1.上传并解压hive的tar包（嫌名字太长就修改一下名字）

2.配置环境变量

3.解决日志jar包冲突，将hive自带的日志jar包删除。（slf4j）

4.在conf目录中新建一个配置文件，hive-site.xml并添加以下配置：

<?xml version="1.0"?>

<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<configuration>

<!-- jdbc连接的URL -->

<property>

<name>javax.jdo.option.ConnectionURL</name>

<value>jdbc:mysql://hadoop101:3306/metastore?useSSL=false</value>

</property>

<!-- jdbc连接的Driver-->

<property>

<name>javax.jdo.option.ConnectionDriverName</name>

<value>com.mysql.jdbc.Driver</value>

</property>

<!-- jdbc连接的username-->

<property>

<name>javax.jdo.option.ConnectionUserName</name>

<value>root</value>

</property>

<!-- jdbc连接的password -->

<property>

<name>javax.jdo.option.ConnectionPassword</name>

<value>123456</value>

</property>

<!-- Hive默认在HDFS的工作目录（存放路径） -->

<property>

<name>hive.metastore.warehouse.dir</name>

<value>/user/hive/warehouse</value>

</property>

<!-- Hive元数据存储的验证 -->

<property>

<name>hive.metastore.schema.verification</name>

<value>false</value>

</property>

<!-- 元数据存储授权 -->

<property>

<name>hive.metastore.event.db.notification.api.auth</name>

<value>false</value>

</property>

</configuration>

由于hive要连接mysql是需要通过jdbc的方式，所以需要jdbc的驱动包，hive通过jdbc的方式连接mysql，所以需要将驱动包放到lib目录下hive才能找得着。

## 完善Hive

初始化原数据库

* 1. **登陆MySQL**
  2. **新建Hive元数据库表空间metastore**
  3. **初始化Hive元数据库**schematool -initSchema -dbType mysql -verbose

## 启动Hive

* 1. 先启动集群
  2. 客户端选择
     1. 直接敲hive启动普通客户端
     2. 启动相对高级的jdbc客户端
        1. cd到bin目录下，其中有个beeline命令，这个家伙就是基于jdbc协议去操作hive的，但hive正常情况下没有提供jdbc协议让咱们去连接，所以非常时期非常操作，hive提供了第二个服务：hiveserver2，它是一个中间商，它能帮助咱们去操作hive。
        2. 启动中间商：hive --service hiveserver2 ,该方式有个缺陷，会停顿，阻塞，而且不能关闭，关闭的话服务就停了。
        3. 通过beeline命令连接中间商：

beeline -u jdbc:hive2://hadoop101:10000 -n tedu

### （可选）使用元数据服务的方式访问Hive

**1）在hive-site.xml文件中添加如下配置信息**

<!-- 指定存储元数据要连接的地址 -->

<property>

<name>hive.metastore.uris</name>

<value>thrift://hadoop102:9083</value>

</property>

**2）启动metastore**

[tedu@hadoop101 hive]$ hive --service metastore

2020-04-24 16:58:08: Starting Hive Metastore Server

注意: 启动后窗口不能再操作，需打开一个新的shell窗口做别的操作

**3）启动 hive**

[tedu@hadoop101 hive]$ bin/hive

脚本相关：

前台启动的方式导致需要打开多个shell窗口，可以使用如下方式后台方式启动

nohup: 放在命令开头，表示不挂起,也就是关闭终端进程也继续保持运行状态

0:标准输入

1:标准输出

2:错误输出

2>&1 : 表示将错误重定向到标准输出上

&: 放在命令结尾,表示后台运行

一般会组合使用: nohup [xxx命令操作]> file 2>&1 & ， 表示将xxx命令运行的

结果输出到file中，并保持命令启动的进程在后台运行。

nohup hive --service hiveserver2 2>&1 &（不要求掌握）

完整脚本：

#!/bin/bash

HIVE\_LOG\_DIR=$HIVE\_HOME/logs

if [ ! -d $HIVE\_LOG\_DIR ]

then

mkdir -p $HIVE\_LOG\_DIR

fi

#检查进程是否运行正常，参数1为进程名，参数2为进程端口

function check\_process()

{

pid=$(ps -ef 2>/dev/null | grep -v grep | grep -i $1 | awk '{print $2}')

ppid=$(netstat -nltp 2>/dev/null | grep $2 | awk '{print $7}' | cut -d '/' -f 1)

echo $pid

[[ "$pid" =~ "$ppid" ]] && [ "$ppid" ] && return 0 || return 1

}

function hive\_start()

{

metapid=$(check\_process HiveMetastore 9083)

cmd="nohup hive --service metastore >$HIVE\_LOG\_DIR/metastore.log 2>&1 &"

cmd=$cmd" sleep 4; hdfs dfsadmin -safemode wait >/dev/null 2>&1"

[ -z "$metapid" ] && eval $cmd || echo "Metastroe服务已启动"

server2pid=$(check\_process HiveServer2 10000)

cmd="nohup hive --service hiveserver2 >$HIVE\_LOG\_DIR/hiveServer2.log 2>&1 &"

[ -z "$server2pid" ] && eval $cmd || echo "HiveServer2服务已启动"

}

function hive\_stop()

{

metapid=$(check\_process HiveMetastore 9083)

[ "$metapid" ] && kill $metapid || echo "Metastore服务未启动"

server2pid=$(check\_process HiveServer2 10000)

[ "$server2pid" ] && kill $server2pid || echo "HiveServer2服务未启动"

}

case $1 in

"start")

hive\_start

;;

"stop")

hive\_stop

;;

"restart")

hive\_stop

sleep 2

hive\_start

;;

"status")

check\_process HiveMetastore 9083 >/dev/null && echo "Metastore服务运行正常" || echo "Metastore服务运行异常"

check\_process HiveServer2 10000 >/dev/null && echo "HiveServer2服务运行正常" || echo "HiveServer2服务运行异常"

;;

\*)

echo Invalid Args!

echo 'Usage: '$(basename $0)' start|stop|restart|status'

;;

esac

给脚本添加执行权限。

**启动Hive后台服务**hiveservices start

## 5. Hive常用交互命令

**1.命令hive -help可以查看hive全部的交互命令**

**(1) “-e”不进入hive的交互窗口执行sql语句**

**(2) “-f”执行脚本中sql语句**

**1.创建一个空文件并写入正确命令**

**2.hive -f 文件名**

## 6. Hive的其他命令操作

**1）退出hive窗口：**

hive(default)>exit;

hive(default)>quit;

在新版的hive中没区别了，在以前的版本是有的：

exit:先隐性提交数据，再退出；

quit:不提交数据，退出；

**2）在hive cli命令窗口中如何查看hdfs文件系统**

hive(default)>dfs -ls /;

**3）查看在hive中输入的所有历史命令**

（1）进入到当前用户的根目录/root或/home/tedu

（2）查看. hivehistory文件

[tedu@hadoop101 ~]$ cat .hivehistory

## Hive常见属性配置

* + 1. hive窗口打印当前库和表头

在hive-site.xml中加入如下两个配置:

<property>

<name>hive.cli.print.header</name>

<value>true</value>

</property>

<property>

<name>hive.cli.print.current.db</name>

<value>true</value>

</property>

* + 1. Hive运行日志信息配置
       1. **Hive的log默认存放在/tmp/tedu/hive.log目录下（当前用户名下）**
       2. **修改hive的log存放日志到/opt/module/hive/logs**
          1. 修改/opt/module/hive/conf/hive-log4j2.properties.template文件名称为hive-log4j2.properties
          2. 在hive-log4j.properties文件中修改log存放位置

property.hive.log.dir=/opt/module/hive/logs

## 参数配置方式

**1）查看当前所有的配置信息**

hive>set;

**2）参数的配置三种方式**

（1）配置文件方式

默认配置文件：hive-default.xml

用户自定义配置文件：hive-site.xml

注意：用户自定义配置会覆盖默认配置。另外，Hive也会读入Hadoop的配置，因为Hive是作为Hadoop的客户端启动的，Hive的配置会覆盖Hadoop的配置。配置文件的设定对本机启动的所有Hive进程都有效。

（2）命令行参数方式

启动Hive时，可以在命令行添加-hiveconf param=value来设定参数。

例如：

[tedu@hadoop101 hive]$ bin/hive -hiveconf mapred.reduce.tasks=10;

注意：仅对本次hive启动有效

查看参数设置：

hive (default)> set mapred.reduce.tasks;

（3）参数声明方式

可以在HQL中使用SET关键字设定参数

例如：

hive (default)> set mapred.reduce.tasks=100;

注意：仅对本次hive启动有效。

## Hive数据类型

## 基本数据类型

|  |  |  |  |
| --- | --- | --- | --- |
| Hive数据类型 | Java数据类型 | 长度 | 例子 |
| TINYINT | byte | 1byte有符号整数 | 20 |
| SMALINT | short | 2byte有符号整数 | 20 |
| INT | int | 4byte有符号整数 | 20 |
| BIGINT | long | 8byte有符号整数 | 20 |
| BOOLEAN | boolean | 布尔类型，true或者false | TRUE FALSE |
| FLOAT | float | 单精度浮点数 | 3.14159 |
| DOUBLE | double | 双精度浮点数 | 3.14159 |
| STRING | string | 字符系列。可以指定字符集。可以使用单引号或者双引号。 | ‘now is the time’ “for all good men” |
| TIMESTAMP |  | 时间类型 |  |
| BINARY |  | 字节数组 |  |

对于Hive的String类型相当于数据库的varchar类型，该类型是一个可变的字符串，不过它不能声明其中最多能存储多少个字符，理论上它可以存储2GB的字符数。

## 集合数据类型

|  |  |  |
| --- | --- | --- |
| 数据类型 | 描述 | 语法示例 |
| STRUCT | 和c语言中的struct类似，都可以通过“点”符号访问元素内容。例如，如果某个列的数据类型是STRUCT{first STRING, last STRING},那么第1个元素可以通过字段.first来引用。 | struct()  例如struct<street:string, city:string> |
| MAP | MAP是一组键-值对元组集合，使用数组表示法可以访问数据。例如，如果某个列的数据类型是MAP，其中键->值对是’first’->’John’和’last’->’Doe’，那么可以通过字段名[‘last’]获取最后一个元素 | map()  例如map<string, int> |
| ARRAY | 数组是一组具有相同类型和名称的变量的集合。这些变量称为数组的元素，每个数组元素都有一个编号，编号从零开始。例如，数组值为[‘John’, ‘Doe’]，那么第2个元素可以通过数组名[1]进行引用。 | Array()  例如array<string> |

Hive有三种复杂数据类型ARRAY、MAP 和 STRUCT。ARRAY和MAP与Java中的Array和Map类似，而STRUCT与C语言中的Struct类似，它封装了一个命名字段集合，复杂数据类型允许任意层次的嵌套。

**1）案例实操**

（1）假设某表有如下一行，我们用JSON格式来表示其数据结构。在Hive下访问的格式为

{

"name": "songsong",

"friends": ["bingbing" , "lili"] , //列表Array,

"children": { //键值Map,

"xiao song": 19 ,

"xiaoxiao song": 18

}

"address": { //结构Struct,

"street": "hui long guan" ,

"city": "beijing"

}

}

（2）基于上述数据结构，我们在Hive里创建对应的表，并导入数据。

创建本地测试文件test.txt

songsong,bingbing\_lili,xiao song:18\_xiaoxiao song:19,hui long guan\_beijing

yangyang,caicai\_susu,xiao yang:18\_xiaoxiao yang:19,chao yang\_beijing

注意：MAP，STRUCT和ARRAY里的元素间关系都可以用同一个字符表示，这里用“\_”。

（3）Hive上创建测试表test

create table test(

name string,

friends array<string>,

children map<string, int>,

address struct<street:string, city:string>

)

row format delimited fields terminated by ','

collection items terminated by '\_'

map keys terminated by ':'

lines terminated by '\n';

字段解释：

row format delimited fields terminated by ',' -- 列分隔符

collection items terminated by '\_' -- STRUCT的分隔符(数据分割符号)

map keys terminated by ':' -- MAP中的key与value的分隔符

lines terminated by '\n'; -- 行分隔符

（4）导入文本数据到测试表

load data local inpath '/opt/module/hive/datas/test.txt' into table test;

（5）访问三种集合列里的数据，以下分别是ARRAY，MAP，STRUCT的访问方式

hive (default)> select friends[1],children['xiao song'],address.city from test

where name="songsong";

OK

\_c0 \_c1 city

lili 18 beijing

Time taken: 0.076 seconds, Fetched: 1 row(s)

## 10.类型转化

Hive的原子数据类型是可以进行隐式转换的，类似于Java的类型转换，例如某表达式使用INT类型，TINYINT会自动转换为INT类型，但是Hive不会进行反向转化，例如，某表达式使用TINYINT类型，INT不会自动转换为TINYINT类型，它会返回错误，除非使用CAST操作。

**1）隐式类型转换规则如下**

（1）任何整数类型都可以隐式地转换为一个范围更广的类型，如TINYINT可以转换成INT，INT可以转换成BIGINT。

（2）所有整数类型、FLOAT和STRING类型都可以隐式地转换成DOUBLE。

（3）TINYINT、SMALLINT、INT都可以转换为FLOAT。

（4）BOOLEAN类型不可以转换为任何其它的类型。

**2）可以使用CAST操作显示进行数据类型转换**

例如CAST('1' AS INT)将把字符串'1' 转换成整数1；如果强制类型转换失败，如执行CAST('X' AS INT)，表达式返回空值 NULL。

0: jdbc:hive2://hadoop101:10000> select '1'+2, cast('1'as int) + 2;

+------+------+--+

| \_c0 | \_c1 |

+------+------+--+

| 3.0 | 3 |

+------+------+--+