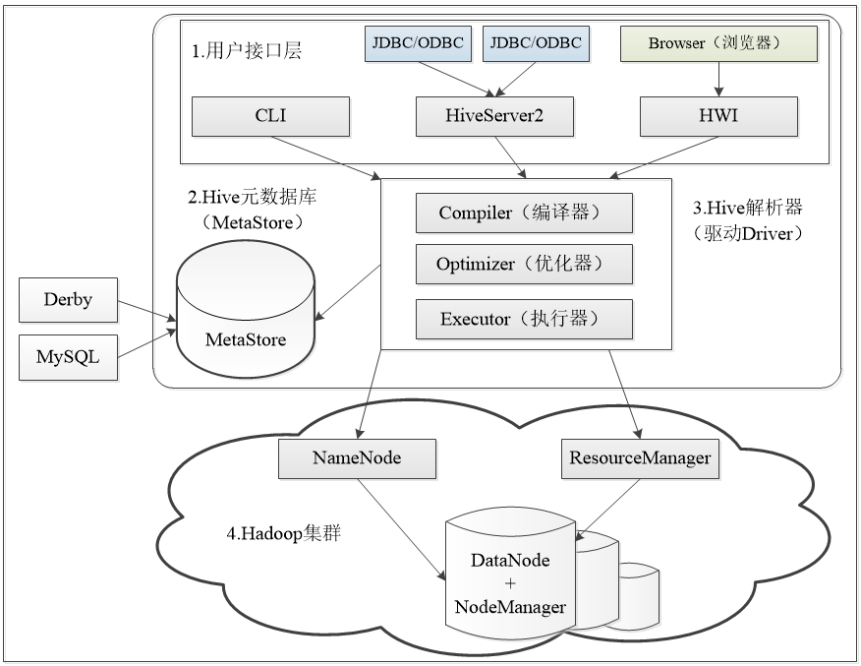
Hive 数据仓库

Hive 数据仓库基于 Hadoop 开发，具备海量数据存储和处理能力，是大数据领域离线批量处理数据的常用工具，是基于 HDFS 和 MapReduce 的分布式数据仓库。Hive 本质上可以理解为一个客户端工具，或是一个将 SQL 语句解析成 MapReduce 作业的引擎，其本身不存储和计算数据，它完全依赖于 HDFS 和 MapReduce。

一、Hive 架构

由用户接口、元数据库、解析器、Hadoop 集群组成，如下图。



1. 用户接口：用于连接访问 Hive，包括命令行接口（CLI）、JDBC/ODBC 和 HWI（Hive Web Interface，Hive Web 接口）3 种方式。
2. Hive 元数据库（MetaStore）：Hive 数据包括数据文件和元数据。数据文件存储在 HDFS 中；元数据存储在数据库中如 Derby（Hive 默认数据库）、MySQL 数据库等，Hive 中的元数据包括表的名字、表的列和分区、表的属性、表的数据所在的目录等。
3. Hive 解析器（驱动 Driver）：核心功能是根据用户编写的 SQL 语法匹配出相应的 MapReduce 模板，并形成对应的 MapReduce job 进行执行。Hive 中的解析器在运行时会读取元数据库 MetaStore 中的相关信息。
4. Hadoop 集群：Hive 用 HDFS 进行存储，用 MapReduce 进行计算，也就是说，Hive 数据仓库的数据存储在 HDFS 中，而业务实际分析计算是利用 MapReduce 执行的。

二、Hive 安装配置

说明：在 linux 中新下载 hadoop3.3.1 版本，替换原 hadoop 2.7.7 版本。替换方式：1.将 local 下的 hadoop 文件夹修改为其他名称，将 3.3.1 版本安装包解压并重命名为 hadoop；2.将 /usr/local/hadoop/etc/hadoop 下的 core-site.xml、hdfs-site.xml、mapred-site.xml、yarn-site.xml 重新配置，可将原版本的配置内容复制粘贴即可。

在 Hadoop 集群中配置 MySQL 数据库和配置 Hive 数据仓库。

1. 配置 MySQL 数据库

Hive 会将表中的元数据信息存储在数据库中，但 Hive 的默认数据库 Derby 存在并发性能差的问题，在实际生产环境中适用性较差，因此常常会使用其他数据库作为元数据库。MySQL 是一个开源的关系型数据库管理系统，适合作为存储 Hive 元数据的数据库。

在 Linux 系统中安装 MySQL。

wget：Linux 中的一个下载文件的工具。

下载并安装 Wget：yum install wget

下载 mysql8.x 源：

wget https://repo.mysql.com//mysql80-community-release-el7-3.noarch.rpm

加载本地 yum 源：

yum localinstall mysql80-community-release-el7-3.noarch.rpm

查询是否存在 mysql-community-server.x86_64：yum search mysql

```
root@host1:/mysql
  名称和简介匹配 only，使用“search all”试试。
[root@host1 mysql]# yum search mysql
已加载插件：fastestmirror
Loading mirror speeds from cached hostfile
 * base: mirrors.jlu.edu.cn
 * extras: mirrors.jlu.edu.cn
 * updates: mirrors.jlu.edu.cn
================================== N/S matched: mysql ==================================
MySQL-python.x86_64 : An interface to MySQL
akonadi-mysql.x86_64 : Akonadi MySQL backend support
apr-util-mysql.x86_64 : APR utility library MySQL DBD driver
dovecot-mysql.x86_64 : MySQL back end for dovecot
freeradius-mysql.x86_64 : MySQL support for freeradius
libdbi-dbd-mysql.x86_64 : MySQL plugin for libdbi
mysql-community-client.i686 : MySQL database client applications and tools
mysql-community-client.x86_64 : MySQL database client applications and tools
mysql-community-client-plugins.i686 : Shared plugins for MySQL client applications
mysql-community-client-plugins.x86_64 : Shared plugins for MySQL client applications
mysql-community-common.i686 : MySQL database common files for server and client libs
mysql-community-common.x86_64 : MySQL database common files for server and client libs
mysql-community-devel.i686 : Development header files and libraries for MySQL database client
                           : applications
mysql-community-devel.x86_64 : Development header files and libraries for MySQL database client
                             : applications
mysql-community-embedded-compat.i686 : MySQL embedded compat library
mysql-community-embedded-compat.x86_64 : MySQL embedded compat library
mysql-community-icu-data-files.i686 : MySQL packaging of ICU data files
mysql-community-icu-data-files.x86_64 : MySQL packaging of ICU data files
mysql-community-libs.i686 : Shared libraries for MySQL database client applications
```

下载 mysql 服务：

yum install mysql-community-server.x86_64 –nogpgcheck

启动 mysql：service mysqld start

```
[root@host1 /]# service mysqld start
Redirecting to /bin/systemctl start mysqld.service
```

查看 mysql 状态：service mysqld status

```
[root@host1 /]# service mysqld status
Redirecting to /bin/systemctl status mysqld.service
● mysqld.service - MySQL Server
   Loaded: loaded (/usr/lib/systemd/system/mysqld.service; enabled; vendor preset: disabled)
   Active: active (running) since 一 2023-05-08 00:52:04 CST; 30s ago
     Docs: man:mysqld(8)
           http://dev.mysql.com/doc/refman/en/using-systemd.html
  Process: 3506 ExecStartPre=/usr/bin/mysqld_pre_systemd (code=exited, status=0/SUCCESS)
 Main PID: 3530 (mysqld)
   Status: "Server is operational"
   CGroup: /system.slice/mysqld.service
           └─3530 /usr/sbin/mysqld

5月 08 00:52:03 host1 systemd[1]: Starting MySQL Server...
5月 08 00:52:04 host1 systemd[1]: Started MySQL Server.
```

查询 mysql 初始密码，冒号空格后面为密码：

cat /var/log/mysqld.log | grep password

```
[root@host1 /]# cat /var/log/mysqld.log | grep password
2023-05-07T16:48:40.789363Z 6 [Note] [MY-010454] [Server] A temporary password is generated for root@loc
alhost: aghqPu2?Ly:P
```

使用初始密码登录，在跳出对话框中粘贴刚刚复制的密码：

mysql -u root -p

```
[root@host1 /]# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.33

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

设置自定义密码：

alter user 'root'@'localhost' identified by '2023%%DtSy';

MySQL 设置远程访问权限命令：

use mysql;

update user set host='%' where user='root';

grant all privileges on *.* to 'root'@'%';

grant all privileges on *.* to 'root'@'%';

刷新权限：flush privileges;

设置完毕退出 mysql 数据库：exit;

2. 配置 Hive 数据仓库

下载 Hive 安装包和 MySQL 驱动包并上传至/usr/local/src 内

下载地址：

http://archive.apache.org/dist/hive/hive-3.1.2/

https://repo.maven.apache.org/maven2/mysql/mysql-connector-java/8.0.20/mysql-connector-java-8.0.20.jar

转至/usr/local/src 内：

cd /usr/local/src

```
[root@host1 ~]# cd /usr/local/src
[root@host1 src]# ls
apache-hive-3.1.2-bin.tar.gz
```

解压至/usr/local 内：

tar -xzvf apache-hive-3.1.2-bin.tar.gz -C /usr/local/

```
[root@host1 local]# ls
apache-hive-3.1.2-bin  bin  etc  games  include  lib  lib64  libexec  sbin  share  src
[root@host1 local]#
```

在命令行输入

mv /usr/local/apache-hive-3.1.2-bin hive

修改 apache-hive-3.1.2-bin 名称为 hive

```
[root@host1 local]# ls
bin  etc  games  hive  include  lib  lib64  libexec  sbin  share  src
[root@host1 local]#
```

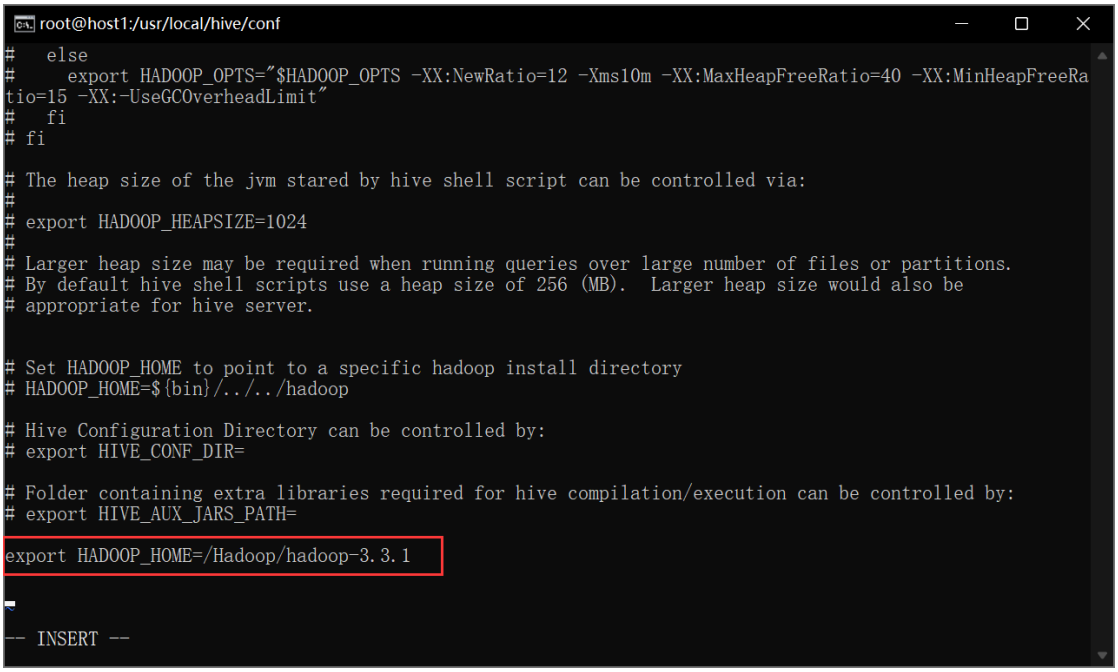
进入/usr/local/hive/conf 目录下

cd /usr/local/hive/conf

cp hive-env.sh.template hive-env.sh

vi hive-env.sh

在当前文件末尾加入 hadoop 安装目录的路径，这里请大家按照自身实际安装情况填写路径。



如果记不清了可以尝试在/etc/profile 上查找(前提是 hadoop 可以正常使用)，或者执行 find / -name hadoop 进行查找

在/etc/profile 加入 hive 环境变量

vi /etc/profile

在已有的 PATH 变量后面增加$HIVE_HOME/bin，若没有 PATH 则添加

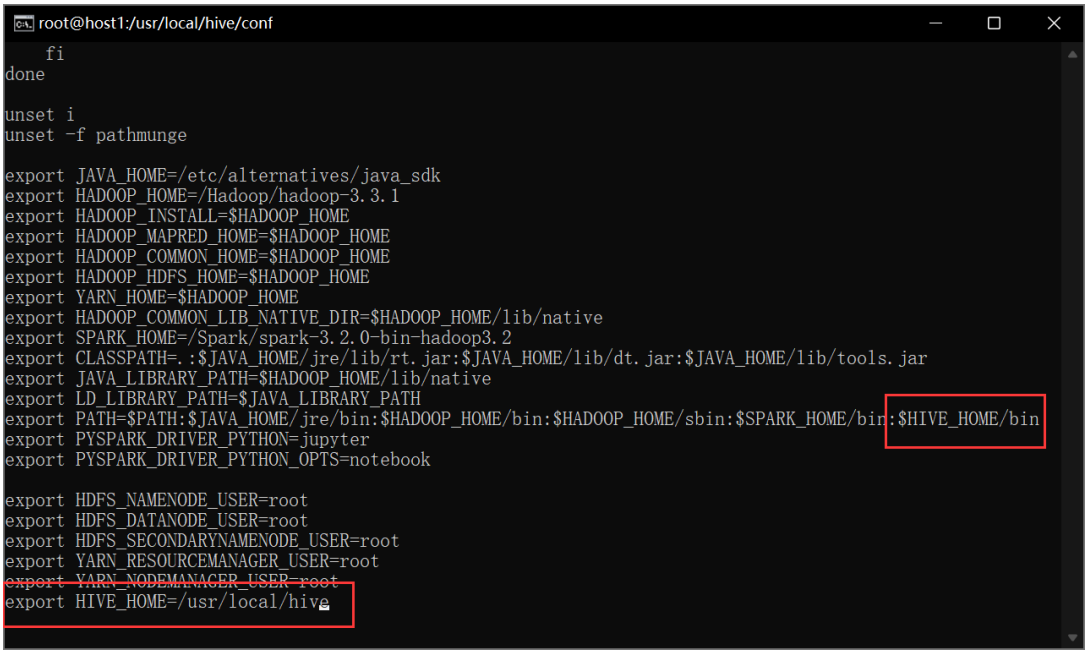
export PATH=$HIVE_HOME/bin:$PATH

至尾端

在当前文件添加

export HIVE_HOME=/usr/local/hive

注意，如果你将 hive 解压到了其他文件夹内请填写 hive 真正所在的文件夹路径



source /etc/profile

在/usr/local/hive/conf 目录下，新建一个名为 hive-site.xml 的文件

vi /usr/local/hive/conf/hive-site.xml

将下列代码复制到该文件内

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
        <property>
          <name>hive.exec.scratchdir</name>
          <value>hdfs://master:8020/user/hive/tmp</value>
        </property>
        <property>
          <name>hive.metastore.warehouse.dir</name>
          <value>hdfs://master:8020/user/hive/warehouse</value>
        </property>
        <property>
          <name>hive.querylog.location</name>
          <value>hdfs://master:8020/user/hive/log</value>
        </property>
        <property>
          <name>hive.metastore.uris</name>
          <value>thrift://master:9083</value>
        </property>
        <property>
          <name>javax.jdo.option.ConnectionURL</name>

<value>jdbc:mysql://master:3306/hive?createDatabaseIfNotExist=true&amp;amp;characterEncoding=UTF-8
&amp;amp;useSSL=false&amp;amp;allowPublicKeyRetrieval=true</value>
        </property>
        <property>
          <name>javax.jdo.option.ConnectionDriverName</name>
          <value>com.mysql.cj.jdbc.Driver</value>
        </property>
        <property>
          <name>javax.jdo.option.ConnectionUserName</name>
          <value>root</value>
        </property>
        <property>
          <name>javax.jdo.option.ConnectionPassword</name>
          <value>2023%%DtSy</value>
        </property>
      <property>
        <name>hive.metastore.schema.verification</name>
        <value>false</value>
```

```
  </property>
<property>
      <name>datanucleus.schema.autoCreateAll</name>
      <value>true</value>
 </property>
 <property>
      <name>hive.support.concurrency</name>
      <value>true</value>
</property>
<property>
      <name>hive.txn.manager</name>
      <value>org.apache.hadoop.hive.ql.lockmgr.DbTxnManager</value>
</property>
</configuration>
```

注意上述文件内密码要改成自己的密码，如果密码和教程密码一样可以直接复制

进入$HIVE_HOME/lib 目录：

```
cd $HIVE_HOME/lib
```

将 MySQL 驱动包复制到$HIVE_HOME/lib 目录中：

```
cp /usr/local/src/mysql-connector-java-8.0.20.jar $HIVE_HOME/lib/
```

更改 guava jar 包版本

移除 Hive 自带的 guava-19.0.jar

```
rm -rf /usr/local/hive/lib/guava-19.0.jar
```

将 hadoop 自带的 guava-27.0-jre.jar 配置到 hive 上

```
cp $HADOOP_HOME/share/hadoop/common/lib/guava-27.0-jre.jar $HIVE_HOME/lib/
find $HIVE_HOME/lib/ -name guava-27.0-jre.jar
```

```
[root@host1 lib]# find $HIVE_HOME/lib/ -name guava-27.0-jre.jar
/usr/local/hive/lib/guava-27.0-jre.jar
[root@host1 lib]#
```

初始化 hive

```
mv /usr/local/hive/lib/log4j-slf4j-impl-2.10.0.jar /usr/local/hive/lib/log4j-slf4j-impl-2.10.0.jar.bak
source /etc/profile
schematool -dbType mysql -initSchema
```

启动 hive(启动之前需要先启动 hadoop 和 mysql)

```
hive --service metastore &
hive
```

3. HiveQL 查询语句

3.1 创建数据表

创建数据库：

```
hive> CREATE DATABASE card;
OK
Time taken: 0.089 seconds
```

创建内部表 info：

首先，进入数据库 card

```
hive> use card;
OK
Time taken: 0.039 seconds
```

在数据库 card 下创建成绩信息表 score 和 info

```
hive> create table score(
    > stu_no string,
    > cla_no string,
    > grade float
    > ) partitioned by (class_name string);
OK
Time taken: 0.056 seconds
```

```
hive> create table info(
    > index int,
    > cardno int,
    > peono int,
    > consumdate string,
    > money float,
    > fundmoney int,
    > surplus float,
    > cardcount int,
    > type string,
    > termserno string,
    > conoperno string,
    > dept string)
    > row format delimited fields terminated by ',';
OK
Time taken: 0.284 seconds
```

将表 score 重命名成 stu_score

```
hive> alter table score rename to stu_score;
OK
Time taken: 0.19 seconds
```

添加列

```
hive> alter table stu_score add columns (credit int,gpa float);
OK
Time taken: 0.082 seconds
```

新增分区

```
hive> alter table stu_score add partition(class_name='07111301');
OK
Time taken: 0.178 seconds
```

删除分区

```
hive> alter table stu_score drop if exists partition(class_name = '07111301');
Dropped the partition class_name=07111301
OK
Time taken: 0.848 seconds
```

3.2 Hive 表的数据装载

将文件系统的数据导入 Hive 表

首先将数据 data1.csv 上传至/data 内，data2.csv 上传至 HDFS 的/user/root/data 内

将本地文件系统中的 data1.csv 导入表 student

```
hive> load data local inpath '/data/data1.csv' overwrite into table student;
Loading data to table card.student
OK
Time taken: 0.841 seconds
```

将 HDFS 中的 data2.csv 导入至表 info

```
hive> load data inpath '/user/root/data/data2.csv' overwrite into table info;
Loading data to table card.info
OK
Time taken: 0.513 seconds
```

3.3 掌握 select 查询

由于 SQL 的广泛应用，所以根据 Hive 本身的特性设计了类 SQL 的查询语言 HQL。HQL 查询语句 select 的语法如下：

```
select [all | distinct] select_expr, select_expr, ...
from table_reference
[where where_condition]
[group by col_list]
[having having_condition]
[order by col_list]
[cluster by col_list | [distribute by col_list] [sort by col_list]]
[limit [offset,] rows]
```

group by：表示根据某个字段对数据进行分组，一般情况下，group by 必须要配合聚合函数（如 count()、max()等）一起使用，以实现在分组之后对组内结果的聚合操作。

order by：使用该关键字可以令查询结果按照某个字段进行排序，默认的情况下为升序（ASC）排序。用户也可以使用 DESC 关键字对查询结果进行降序排序。

limit： limit 关键字可用于约束查询结果返回的行数。

例如：查看 info 表中的 cardno、消费类型为消费的消费地点数据

```
182704  消费    第四食堂
182704  消费    第二食堂
182704  消费    第一食堂
182705  消费    好利来食品店
182705  消费    好利来食品店
182705  消费    第三食堂
182705  消费    第三食堂
182705  消费    第三食堂
182705  消费    第五食堂
182705  消费    第五食堂
182705  消费    第五食堂
182705  消费    第五食堂
182706  消费    第三食堂
182706  消费    第四食堂
182706  消费    第五食堂
182706  消费    第五食堂
182706  消费    第五食堂
182706  消费    第五食堂
182706  消费    第五食堂
182706  消费    第五食堂
182707  消费    第三食堂
Time taken: 0.136 seconds, Fetched: 500755 row(s)
hive> select cardno,type,dept from info where type=="消费";
```

数据量大，上面的是部分结果，下面的是代码

3.4 Hive 单表插入数据示例

语法如下：

```
insert [overwrite|into] table  表 1
[partition (part1=val1,part2=val2)]
select 字段 1，字段 2，字段 3   from   表 2 ;
```

该语句表示从表 2 查询出字段 1、字段 2 和字段 3 的数据并插入表 1 中，表 1 中的 3 个字段

的类型与表 2 中的 3 个字段的类型应一致。

```
hive> insert into student_out select index,cardno,sex,major,accesscardno from student;
Query ID = root_20230508182335_d5ca9a60-efa6-4ca9-911e-9ab185feb684
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2023-05-08 18:23:38,843 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local1560489112_0001
Stage-4 is selected by condition resolver.
Stage-3 is filtered out by condition resolver.
Stage-5 is filtered out by condition resolver.
Moving data to directory hdfs://Host2:9000/user/root/data/.hive-staging_hive_2023-05-08_18-23-35_966_403
866772208798053-1/-ext-10000
Loading data to table card.student_out
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 353446 HDFS Write: 344866 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
Time taken: 3.319 seconds
```

3.5 表生成函数

表生成函数可将单个输入行转换为多个输出行。

表生成函数的语法“select udtf（col） as colAlias...”

3.6 聚合函数

聚合函数是对一组值进行计算并返回单一值的函数。聚合函数经常与 select 语句的 group by 子句一同使用。常用的聚合函数如下表。

返回类型	函数	描述
bigint	count(*)，count(expr)，count(DISTINCT expr[, expr...])	count（*）返回检索到的行的总数，包括包含空值的行。 count（expr）返回提供的表达式为非空的行数。 count（DISTINCT expr[，expr]）返回所提供表达式唯一且非空的行数
double	sum(col)，sum(DISTINCT col)	返回组中元素的总和或组中列的不同值的总和
double	avg(col)，avg(DISTINCT col)	返回组中元素的平均值或组中列的不同值的平均值
double	min(col)	返回组中列的最小值
double	max(col)	返回组中列的最大值