

Hive 基础知识

目录

1、Hive 基本概念	1
1.1、Hive 简介	1
1.1.1、什么是 Hive	1
1.1.2、为什么使用 Hive	2
1.1.3、Hive 特点	2
1.2、Hive 和 RDBMS 的对比	3
1.3、Hive 和 HBase 的差别	3
1.4、Hive 架构	4
1.5、Hive 的数据存储	5
2、Hive 环境搭建	6
2.1、Hive 安装	6
2.1.1、内嵌 Derby 版本	6
2.1.2、外置 MySQL 版本	7
2.1.3、Linux RPM 方式安装 MySQL	9
2.2、Hive 使用方式，即三种连接方式	11
2.2.1、CLI	11
2.2.2、HiveServer2/beeline	12
2.2.3、Web UI	14
3、Hive 基本使用	15

1、Hive 基本概念

1.1、Hive 简介

1.1.1、什么是 Hive

Hive 的概念：

- 1、Hive 由 Facebook 实现并开源
- 2、Hive 是基于 Hadoop 的一个数据仓库工具
- 3、Hive 存储的数据其实底层存储在 HDFS 上
- 4、Hive 将 HDFS 上的结构化的数据映射为一张数据库表
- 5、Hive 提供 HQL(Hive SQL)查询功能
- 6、Hive 的本质是将 SQL 语句转换为 MapReduce 任务运行，使不熟悉 MapReduce 的用户很方便地利用 HQL 处理和计算 HDFS 上的结构化的数据，适用于离线的批量数据计算

7、Hive 使用户可以极大简化分布式计算程序的编写，而将精力集中于业务逻辑

数据仓库之父比尔·恩门（Bill Inmon）在 1991 年出版的“Building the Data Warehouse”（《建立数据仓库》）一书中所提出的定义被广泛接受——**数据仓库（Data Warehouse）**是一个面向主题的（**Subject Oriented**）、集成的（**Integrated**）、相对稳定的（**Non-Volatile**）、反映历史变化（**Time Variant**）的数据集合，用于支持管理决策(Decision Making Support)。

Hive 依赖于 HDFS 存储数据，Hive 将 HQL 转换成 MapReduce 执行
所以说 **Hive 是基于 Hadoop 的一个数据仓库工具，实质就是一款基于 HDFS 的 MapReduce 计算框架，对存储在 HDFS 中的数据进行分析和管理的**



1.1.2、为什么使用 Hive

直接使用 MapReduce 所面临的问题：

- 人员学习成本太高
- 项目周期要求太短
- MapReduce 实现复杂查询逻辑开发难度太大

为什么要使用 Hive：

- 更友好的接口：操作接口采用类 SQL 的语法，提供快速开发的能力
- 更低的学习成本：避免了写 MapReduce，减少开发人员的学习成本
- 更好的扩展性：可自由扩展集群规模而无需重启服务，还支持用户自定义函数

1.1.3、Hive 特点

优点：

- 1、**可扩展性,横向扩展**，Hive 可以自由的扩展集群的规模，一般情况下不需要重启服务
 横向扩展：通过分担压力的方式扩展集群的规模
 纵向扩展：一台服务器 cpu i7-6700k 4 核心 8 线程, 8 核心 16 线程, 内存 64G => 128G
- 2、**延展性**，Hive 支持自定义函数，用户可以根据自己的需求来实现自己的函数
- 3、**良好的容错性**，可以保障即使有节点出现问题，SQL 语句仍可完成执行

缺点：

- 1、**Hive 不支持记录级别的增删改操作**，但是用户可以通过查询生成新表或者将查询结果导入到文件中（当前选择的 hive-2.3.3 的版本支持记录级别的插入操作）
- 2、**Hive 的查询延时很严重**，因为 MapReduce Job 的启动过程消耗很长时间，所以不能用在交互查询系统中。
- 3、**Hive 不支持事务**（因为没有增删改，所以主要用来做 OLAP（联机分析处理），而不是 OLTP（联机事务处理），这就是数据处理的两大级别）。

1.2、Hive 和 RDBMS 的对比

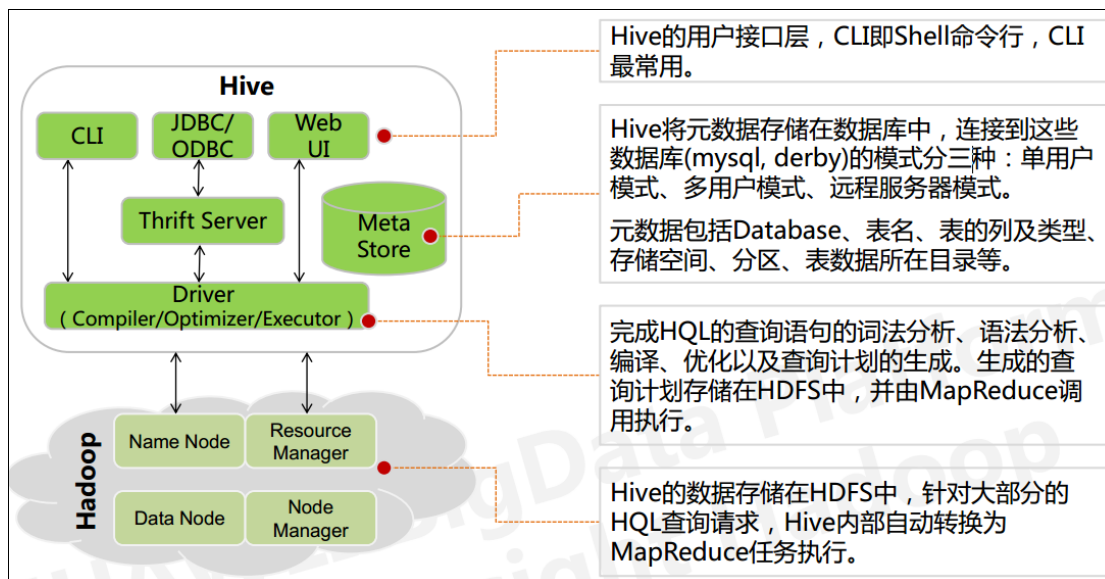
对比项	Hive	RDBMS
查询语言	HQL	SQL
数据存储	HDFS	Raw Device or Local FS
执行器	MapReduce	Executor
数据插入	支持批量导入/单条插入	支持单条或者批量导入
数据操作	覆盖追加	行级更新删除
处理数据规模	大	小
执行延迟	高	低
分区	支持	支持
索引	0.8 版本之后加入简单索引	支持复杂的索引
扩展性	高（好）	有限（差）
数据加载模式	读时模式（快）	写时模式（慢）
应用场景	海量数据查询	实时查询

总结：Hive 具有 SQL 数据库的外表，但应用场景完全不同，**Hive 只适合用来做海量离线数据统计分析，也就是数据仓库。**

1.3、Hive 和 HBase 的差别

- 1、Hive 是建立在 Hadoop 之上为了降低 MapReduce 编程复杂度的 ETL 工具。
HBase 是为了弥补 Hadoop 对实时操作的缺陷
- 2、Hive 表是纯逻辑表，因为 Hive 的本身并不能做数据存储和计算，而是完全依赖 Hadoop
HBase 是物理表，提供了一张超大的内存 Hash 表来存储索引，方便查询
- 3、Hive 是数据仓库工具，需要全表扫描，就用 Hive，因为 Hive 是文件存储
HBase 是数据库，需要索引访问，则用 HBase，因为 HBase 是面向列的 NoSQL 数据库
- 4、Hive 表中存入数据（文件）时不做校验，属于读模式存储系统
HBase 表插入数据时，会和 RDBMS 一样做 Schema 校验，所以属于写模式存储系统
- 5、Hive 不支持单行记录操作，数据处理依靠 MapReduce，操作延时高
HBase 支持单行记录的 CRUD，并且是实时处理，效率比 Hive 高得多

1.4、Hive 架构



基本组成

一、用户接口

CLI, Shell 终端命令行 (Command Line Interface), 采用交互形式使用 Hive 命令行与 Hive 进行交互, 最常用 (学习, 调试, 生产)

JDBC/ODBC, 是 Hive 的基于 JDBC 操作提供的客户端, 用户 (开发员, 运维人员) 通过这连接至 Hive server 服务

Web UI, 通过浏览器访问 Hive

二、Thrift Server

Thrift 是 Facebook 开发的一个软件框架, 可以用来进行可扩展且跨语言的服务的开发, Hive 集成了该服务, 能让不同的编程语言调用 Hive 的接口

三、元数据存储

元数据, 通俗的讲, 就是存储在 Hive 中的数据的描述信息。

Hive 中的**元数据**通常包括: 表的名字, 表的列和分区及其属性, 表的属性 (内部表和外部表), 表的数据所在目录

Metastore 默认存在自带的 Derby 数据库中。缺点就是不适合多用户操作, 并且数据存储目录不固定。数据库跟着 Hive 走, 极度不方便管理

解决方案: 通常存我们自己创建的 MySQL 库 (本地 或 远程)

Hive 和 MySQL 之间通过 MetaStore 服务交互

四、Driver: 编译器 (Compiler), 优化器 (Optimizer), 执行器 (Executor)

Driver 组件完成 HQL 查询语句从词法分析, 语法分析, 编译, 优化, 以及生成逻辑执行计划的生成。生成的逻辑执行计划存储在 HDFS 中, 并随后由 MapReduce 调用执行

Hive 的核心是驱动引擎, 驱动引擎由四部分组成:

(1) 解释器: 解释器的作用是将 HiveSQL 语句转换为抽象语法树 (AST)

- (2) 编译器：编译器是将语法树编译为逻辑执行计划
- (3) 优化器：优化器是对逻辑执行计划进行优化
- (4) 执行器：执行器是调用底层的运行框架执行逻辑执行计划

五、执行流程

HiveQL 通过命令行或者客户端提交，经过 Compiler 编译器，运用 MetaStore 中的元数据进行类型检测和语法分析，生成一个逻辑方案(Logical Plan)，然后通过的优化处理，产生一个 MapReduce 任务。

1.5、Hive 的数据存储

1、Hive 的存储结构包括**数据库、表、视图、分区和表数据**等。数据库，表，分区等等都对应 HDFS 上的一个目录。表数据对应 HDFS 对应目录下的文件。

2、Hive 中所有的数据都存储在 HDFS 中，没有专门的数据存储格式，因为 Hive 是**读模式** (Schema On Read)，可支持 TextFile，SequenceFile，RCFile 或者自定义格式等

3、只需要在创建表的时候告诉 Hive 数据中的**列分隔符**和**行分隔符**，Hive 就可以解析数据
Hive 的默认列分隔符：控制符 Ctrl + A, \x01
Hive 的默认行分隔符：换行符 \n

4、Hive 中包含以下数据模型：

- database**：在 HDFS 中表现为\${hive.metastore.warehouse.dir}目录下一个文件夹
- table**：在 HDFS 中表现所属 database 目录下一个文件夹
- external table**：与 table 类似，不过其数据存放位置可以指定任意 HDFS 目录路径
- partition**：在 HDFS 中表现为 table 目录下的子目录
- bucket**：在 HDFS 中表现为同一个表目录或者分区目录下根据某个字段的值进行 hash 散列之后的多个文件
- view**：与传统数据库类似，只读，基于基本表创建

5、Hive 的元数据存储于 RDBMS 中，除元数据外的其它所有数据都基于 HDFS 存储。默认情况下，Hive 元数据保存在内嵌的 Derby 数据库中，只能允许一个会话连接，只适合简单的测试。实际生产环境中不适用，为了支持多用户会话，则需要一个独立的元数据库，使用 MySQL 作为元数据库，Hive 内部对 MySQL 提供了很好的支持。

6、Hive 中的表分为**内部表、外部表、分区表**和 **Bucket 表**

内部表和外部表的区别：

- 删除内部表，删除表元数据和数据
- 删除外部表，删除元数据，不删除数据

内部表和外部表的使用选择：

大多数情况，他们的区别不明显，如果数据的所有处理都在 Hive 中进行，那么倾向于选择内部表，但是如果 Hive 和其他工具要针对相同的数据集进行处理，外部表更合适。

使用外部表访问存储在 HDFS 上的初始数据，然后通过 Hive 转换数据并存到内部表中
使用外部表的场景是针对一个数据集有多个不同的 Schema

通过外部表和内部表的区别和使用选择的对比可以看出来，hive 其实仅仅只是对存储在 HDFS 上的数据提供了一种新的抽象。而不是管理存储在 HDFS 上的数据。所以不管创建内部表还是外部表，都可以对 hive 表的数据存储目录中的数据进行增删操作。

分区表和分桶表的区别：

Hive 数据表可以根据某些字段进行分区操作，细化数据管理，可以让部分查询更快。同时表和分区也可以进一步被划分为 Buckets，分桶表的原理和 MapReduce 编程中的 HashPartitioner 的原理类似

分区和分桶都是细化数据管理，但是分区表是手动添加区分，由于 Hive 是读模式，所以对添加进分区的数据不做模式校验，分桶表中的数据是按照某些分桶字段进行 hash 散列形成的多个文件，所以数据的准确性也高很多

关于分桶表和分区表的区别，具体详见文档：[Hive 分区和分桶补充---笔记.txt](#)

2、Hive 环境搭建

2.1、Hive 安装

2.1.1、内嵌 Derby 版本

- 1、上传安装包 `apache-hive-2.3.3-bin.tar.gz`
- 2、解压安装包 `tar -zxvf apache-hive-2.3.3-bin.tar.gz -C /home/hadoop/apps/`
- 3、进入到 bin 目录，运行 hive 脚本：`[hadoop@hadoop02 bin]$./hive`

注意：

1、这时候一般会报错：Terminal initialization failed; falling back to unsupported，是因为 hadoop（\$HADOOP_HOME/share/hadoop/yarn/lib）集群的 jline-0.9.94.jar 包版本过低，替换成\$HIVE_HOME/lib 中的 jline-2.12.jar 包即可。记住：所有 hdfs 节点都得替换 \$HADOOP_HOME/share/hadoop/yarn/lib/jline-0.9.4.jar 替换成 jline-2.12.jar

2、修改 log4j.properties（如果有关于日志报错，请照此修改）

`cp hive-log4j.properties.template hive-log4j.properties`

将 EventCounter 修改成 org.apache.hadoop.log.metrics.EventCounter

`#log4j.appender.EventCounter=org.apache.hadoop.hive.shims.HiveEventCounter`

`log4j.appender.EventCounter=org.apache.hadoop.log.metrics.EventCounter`

如果报错就按照此方式解决，没有报错就不用管，在使用新的 **hadoop-2.7.5** 版本中已经不存在这个问题。所以不用关注。

2.1.2、外置 MySQL 版本

1、准备好 MySQL（请参考以下文档，或者自行安装 MySQL，或者一个可用的 MySQL）

2、上传安装包 **apache-hive-2.3.3-bin.tar.gz**

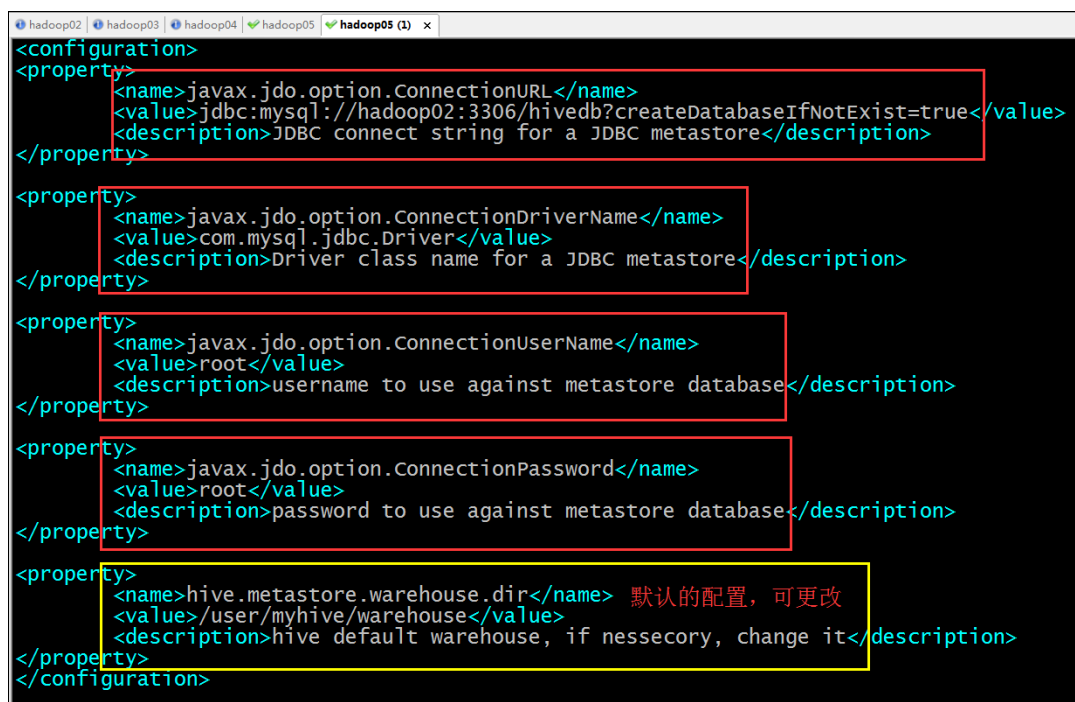
3、解压安装包 **tar -zxvf apache-hive-2.3.3-bin.tar.gz -c ~/apps/**

4、修改配置文件

[hadoop@hadoop02 conf]# **touch hive-site.xml**

[hadoop@hadoop02 conf]# **vi hive-site.xml**

在这个新创建的配置文件中加入如下截图中的内容即可：



```
<configuration>
<property>
  <name>javax.jdo.option.ConnectionURL</name>
  <value>jdbc:mysql://hadoop02:3306/hivedb?createDatabaseIfNotExist=true</value>
  <description>JDBC connect string for a JDBC metastore</description>
</property>
<property>
  <name>javax.jdo.option.ConnectionDriverName</name>
  <value>com.mysql.jdbc.Driver</value>
  <description>Driver class name for a JDBC metastore</description>
</property>
<property>
  <name>javax.jdo.option.ConnectionUserName</name>
  <value>root</value>
  <description>username to use against metastore database</description>
</property>
<property>
  <name>javax.jdo.option.ConnectionPassword</name>
  <value>root</value>
  <description>password to use against metastore database</description>
</property>
<property>
  <name>hive.metastore.warehouse.dir</name> 默认的配置，可更改
  <value>/user/myhive/warehouse</value>
  <description>hive default warehouse, if nessecory, change it</description>
</property>
</configuration>
```

```
<configuration>
  <property>
    <name>javax.jdo.option.ConnectionURL</name>
    <value>jdbc:mysql://hadoop02:3306/hivedb?createDatabaseIfNotExist=true</value>
    <description>JDBC connect string for a JDBC metastore</description>
    <!-- 如果 mysql 和 hive 在同一个服务器节点,那么请更改 hadoop02 为 localhost -->
  </property>

  <property>
    <name>javax.jdo.option.ConnectionDriverName</name>
    <value>com.mysql.jdbc.Driver</value>
    <description>Driver class name for a JDBC metastore</description>
```



```

</property>

<property>
<name>javax.jdo.option.ConnectionUserName</name>
<value>root</value>
<description>username to use against metastore database</description>
</property>

<property>
<name>javax.jdo.option.ConnectionPassword</name>
<value>root</value>
<description>password to use against metastore database</description>
</property>
</configuration>

```

可选配置，该配置信息用来指定 Hive 数据仓库的数据存储在 HDFS 上的目录

```

<property>
<name>hive.metastore.warehouse.dir</name>
<value>/user/hive/warehouse</value>
<description>hive default warehouse, if nessecory, change it</description>
</property>

```

5、一定要记得加入 **MySQL 驱动包** (mysql-connector-java-5.1.40-bin.jar) 该 jar 包放置在 hive 的根路径下的 lib 目录

6、安装完成，配置环境变量

vi ~/.bashrc 添加以下两行内容：

```
export HIVE_HOME=/home/hadoop/apps/apache-hive-2.3.3-bin
```

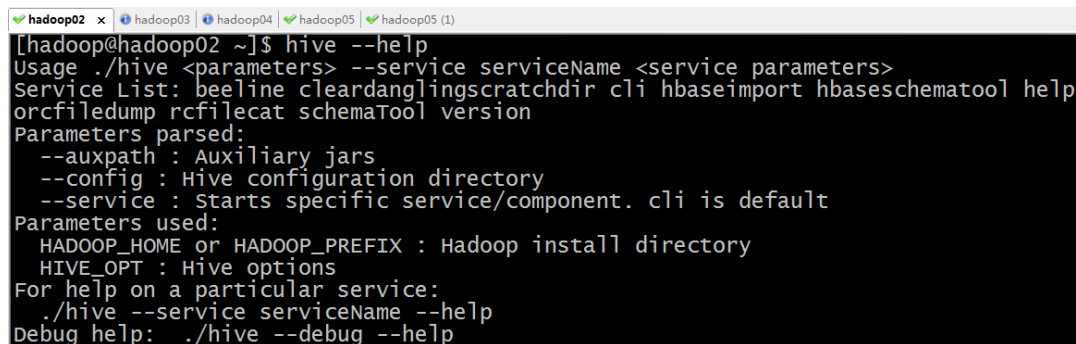
```
export PATH=$PATH:$HIVE_HOME/bin
```

保存退出。

最后不要忘记：[hadoop@hadoop02 bin]\$ **source ~/.bashrc**

7、验证 Hive 安装

```
[hadoop@hadoop02 bin]$ hive --help
```



```

hadoop02 x | hadoop03 | hadoop04 | hadoop05 | hadoop05 (1)
[hadoop@hadoop02 ~]$ hive --help
Usage ./hive <parameters> --service serviceName <service parameters>
Service List: beeline cleardanglingscratchdir cli hbaseimport hbaseschematool help
orcfiledump rcfilecat schemaTool version
Parameters parsed:
  --auxpath : Auxiliary jars
  --config : Hive configuration directory
  --service : Starts specific service/component. cli is default
Parameters used:
  HADOOP_HOME or HADOOP_PREFIX : Hadoop install directory
  HIVE_OPT : Hive options
For help on a particular service:
./hive --service serviceName --help
debug help: ./hive --debug --help

```

8、初始化元数据库

注意：当使用的 hive 是 2.x 之前的版本，不做初始化也是 OK 的，当 hive 第一次启动的时候会自动进行初始化，只不过会不会生成足够多的元数据库中的表。在使用过程中会慢慢生成。但最后进行初始化。如果使用的 2.x 版本的 Hive，那么就必须手动初始化元数据库。使用命令：

```
[hadoop@hadoop02 bin]$ schematool -dbType mysql -initSchema
```

```
hadoop02 x hadoop03 hadoop04 hadoop05
[hadoop@hadoop02 ~]$ source .bashrc
[hadoop@hadoop02 ~]$ schematool -dbType mysql -initSchema
Metastore connection URL: jdbc:mysql://localhost:3306/hive_metadata_db?use
ateDatabaseIfNotExist=true
Metastore Connection Driver : com.mysql.jdbc.Driver
Metastore connection User: root
Starting metastore schema initialization to 1.2.0
Initialization script hive-schema-1.2.0.mysql.sql
Initialization script completed
schematool completed
[hadoop@hadoop02 ~]$
```

9、启动 Hive 客户端

```
[hadoop@hadoop02 bin]$ hive --service cli
```

或者

```
[hadoop@hadoop02 bin]$ hive
```

10、退出 Hive

```
hive> quit;
```

或者

```
hive> exit;
```

2.1.3、Linux RPM 方式安装 MySQL

（记得使用 root 账户进行操作，若使用普通用户，那么请修改相应文件夹权限）

（如果已经有可以使用的 MySQL，不管安装在哪里，只要能远程连接上，都是可以的）

1、检查以前是否装过 MySQL

```
rpm -qa|grep -i mysql
```

结果：

```
[root@hadoop01 ~]# rpm -qa|grep -i mysql
mysql-libs-5.1.73-5.el6_6.x86_64
```

2、发现有的话就都卸载

```
[root@hadoop01 ~]# rpm -qa|grep -i mysql
mysql-libs-5.1.73-5.el6_6.x86_64
[root@hadoop01 ~]# rpm -e --nodeps mysql-libs-5.1.73-5.el6_6.x86_64
[root@hadoop01 ~]#
```

3、删除老版本 mysql 的开发头文件和库

```
rm -rf /usr/lib/mysql #数据库目录
```

```
rm -rf /usr/include/mysql
```

```
rm -f /etc/my.cnf
```

```
rm -rf /var/lib/mysql
```

注意：卸载后/var/lib/mysql 中的数据及/etc/my.cnf 不会删除，确定没用后就手工删除

4、准备安装包 MySQL-5.6.26-1.linux_glibc2.5.x86_64.rpm-bundle.tar， 上传，解压

命令：tar -zxvf MySQL-5.6.26-1.linux_glibc2.5.x86_64.rpm-bundle.tar

```

-rw-r--r-- 1 root root 153530841 Mar 23 2016 glibc-2.17-1.linux.x86_64.tar.gz
-rw-r--r-- 1 root root 317030400 Aug 25 2015 MySQL-5.6.26-1.linux_glibc2.5.x86_64.rpm-bundle.tar
[root@hadoop01 ~]# tar -zxvf MySQL-5.6.26-1.linux_glibc2.5.x86_64.rpm-bundle.tar
gzip: stdin: not in gzip format
tar: Child returned status 1
tar: Error is not recoverable: exiting now
[root@hadoop01 ~]# tar -xvf MySQL-5.6.26-1.linux_glibc2.5.x86_64.rpm-bundle.tar
MySQL-server-5.6.26-1.linux_glibc2.5.x86_64.rpm
MySQL-shared-5.6.26-1.linux_glibc2.5.x86_64.rpm
MySQL-devel-5.6.26-1.linux_glibc2.5.x86_64.rpm
MySQL-client-5.6.26-1.linux_glibc2.5.x86_64.rpm
MySQL-shared-compat-5.6.26-1.linux_glibc2.5.x86_64.rpm
MySQL-embedded-5.6.26-1.linux_glibc2.5.x86_64.rpm
MySQL-test-5.6.26-1.linux_glibc2.5.x86_64.rpm
[root@hadoop01 ~]#

```

5、开始安装

6、安装 server

rpm -ivh MySQL-server-5.6.26-1.linux_glibc2.5.x86_64.rpm

开头：

```

[root@hadoop01 ~]# rpm -ivh MySQL-server-5.6.26-1.linux_glibc2.5.x86_64.rpm
warning: MySQL-server-5.6.26-1.linux_glibc2.5.x86_64.rpm: Header V3 DSA/SHA1 signature, key ID 5072e1f5: NOKEY
Preparing...
1:MySQL-server
warning: user mysql does not exist - using root

```

结尾：

```

Please report any problems at http://bugs.mysql.com/

The latest information about MySQL is available on the web at

http://www.mysql.com

Support MySQL by buying support/licenses at http://shop.mysql.com

New default config file was created as /usr/my.cnf and
will be used by default by the server when you start it.
You may edit this file to change server settings

```

如上图所提示，即安装 server 成功

7、安装客户端

rpm -ivh MySQL-client-5.6.26-1.linux_glibc2.5.x86_64.rpm

```

[root@hadoop01 ~]# rpm -ivh MySQL-client-5.6.26-1.linux_glibc2.5.x86_64.rpm
warning: MySQL-client-5.6.26-1.linux_glibc2.5.x86_64.rpm: Header V3 DSA/SHA1 signature, key ID 5072e1f5: NOKEY
Preparing...
1:MySQL-client

```

8、登陆 MYSQL（登录之前千万记得一定要启动 mysql 服务）

启动命令：

```
[hadoop@hadoop01 ~]$ service mysql start
```

然后登陆，初始密码在 /root/.mysql_secret 这个文件里

```

[root@hadoop01 ~]# cat /root/.mysql_secret
# The random password set for the root user at Thu Nov 3 04:38:15 2016 (local time): CF7y18_Hoq3rkA6x

[root@hadoop01 ~]# mysql -uroot -pCF7y18_Hoq3rkA6x
Warning: Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.6.26

Copyright (c) 2000, 2015, Oracle and/or its affiliates. All rights reserved.

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>

```

9、修改密码

```
set PASSWORD=PASSWORD('root');
```

```
mysql> set PASSWORD=PASSWORD('root');
```

10、退出登陆验证，看是否改密码成功

11、增加远程登陆权限，执行以下两个命令：

```
mysql> GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' IDENTIFIED BY 'root' WITH GRANT OPTION;  
mysql> FLUSH PRIVILEGES;
```

命令释义：grant 权限 1,权限 2,...权限 n on 数据库名称.表名称 to 用户名@用户地址 identified by '密码';

PS: 1,权限 2,...权限 n 代表 select, insert, update, delete, create, drop, index, alter, grant, references, reload, shutdown, process, file 等 14 个权限。

当权限 1,权限 2,...权限 n 被 all privileges 或者 all 代替，表示赋予用户全部权限。

当数据库名称.表名称被 *.* 代替，表示赋予用户操作服务器上所有数据库所有表的权限。

用户地址可以是 localhost，也可以是 ip 地址、机器名字、域名。也可以用 '%' 地址连接。

12、至此 MySQL 安装成功

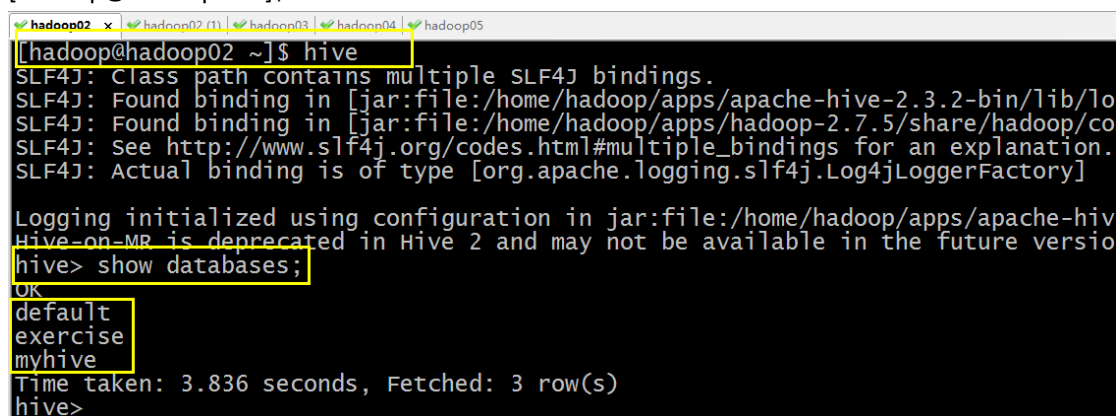
13、更改数据库的默认编码为 UTF-8

2.2、Hive 使用方式，即三种连接方式

2.2.1、CLI

进入到 bin 目录下，直接输入命令：

```
[hadoop@hadoop02 ~]$ hive
```



```
SLF4J: Class path contains multiple SLF4J bindings.  
SLF4J: Found binding in [jar:file:/home/hadoop/apps/apache-hive-2.3.2-bin/lib/lo  
SLF4J: Found binding in [jar:file:/home/hadoop/apps/hadoop-2.7.5/share/hadoop/co  
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.  
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]  
  
Logging initialized using configuration in jar:file:/home/hadoop/apps/apache-hiv  
Hive-on-MR is deprecated in Hive 2 and may not be available in the future versio  
hive> show databases;  
OK  
default  
exercise  
myhive  
Time taken: 3.836 seconds, Fetched: 3 row(s)  
hive>
```

启动成功的话如上图所示，接下来便可以做 hive 相关操作

补充：

1、上面的 hive 命令相当于在启动的时候执行：**hive --service cli**

- 2、使用 **hive --help**，可以查看 hive 命令可以启动那些服务
- 3、通过 **hive --service serviceName --help** 可以查看某个具体命令的使用方式

2.2.2、HiveServer2/beeline

在现在使用的最新的 hive-2.3.2 版本中：都需要对 **hadoop** 集群做如下改变，否则无法使用

第一：修改 hadoop 集群的 hdfs-site.xml 配置文件：加入一条配置信息，表示启用 webhdfs

```
<property>
    <name>dfs.webhdfs.enabled</name>
    <value>true</value>
</property>
```

第二：修改 hadoop 集群的 core-site.xml 配置文件：加入两条配置信息：表示设置 hadoop 的代理用户

```
<property>
    <name>hadoop.proxyuser.hadoop.hosts</name>
    <value>*</value>
</property>
<property>
    <name>hadoop.proxyuser.hadoop.groups</name>
    <value>*</value>
</property>
```

配置解析：

hadoop.proxyuser.hadoop.hosts 配置成*的意义，表示任意节点使用 hadoop 集群的代理用户
hadoop 都能访问 hdfs 集群，hadoop.proxyuser.hadoop.groups 表示代理用户的组所属

以上操作做好了之后，请继续做如下两步：

第一步：先启动 hiveserver2 服务

启动方式，（假如是在 hadoop02 上）：

启动为前台：hiveserver2

启动为后台：

nohup hiveserver2 1>/home/hadoop/hiveserver.log 2>/home/hadoop/hiveserver.err &

或者：nohup hiveserver2 1>/dev/null 2>/dev/null &

或者：nohup hiveserver2 >/dev/null 2>&1 &

以上 3 个命令是等价的，第一个表示记录日志，第二个和第三个表示不记录日志

命令中的 1 和 2 的意义分别是：

1：表示标准日志输出

2：表示错误日志输出

如果我没有配置日志的输出路径，日志会生成在当前工作目录，默认的日志名称叫做：

nohup.xxx

```
hadoop@hadoop02 ~]$ nohup hiveserver2 1>/home/hadoop/hiveserver.log 2>/home/hadoop/hiveserver.err &
[1] 9435
hadoop@hadoop02 ~]$ jps
8354 DataNode
8243 NameNode
9622 Jps
8551 JournalNode
2491 QuorumPeerMain
9435 RunJar
7948 NodeManager
8748 DFSZKFailoverController
hadoop@hadoop02 ~]$
```

PS: nohup 命令：如果你正在运行一个进程，而且你觉得在退出帐户时该进程还会不会结束，那么可以使用 nohup 命令。该命令可以在你退出帐户/关闭终端之后继续运行相应的进程。nohup 就是不挂起的意思(no hang up)。
该命令的一般形式为：nohup command &

第二步：然后启动 beeline 客户端去连接：

执行命令：

beeline -u jdbc:hive2://hadoop02:10000 -n hadoop

-u：指定元数据库的链接信息

-n：指定用户名和密码

```
hadoop@hadoop02 ~]$ beeline -u jdbc:hive2://hadoop02:10000 -n hadoop
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/hadoop/apps/apache-hive-2.3.2-bin/lib/log4j-slf4j-imp]
SLF4J: Found binding in [jar:file:/home/hadoop/apps/hadoop-2.7.5/share/hadoop/common/lib/slf4j]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Connecting to jdbc:hive2://hadoop02:10000
Connected to: Apache Hive (version 2.3.2)
Driver: Hive JDBC (version 2.3.2)
Transaction isolation: TRANSACTION_REPEATABLE_READ
Beeline version 2.3.2 by Apache Hive
0: jdbc:hive2://hadoop02:10000> show databases;
+-----+
| database_name |
+-----+
| default      |
| exercise     |
| myhive       |
+-----+
3 rows selected (1.305 seconds)
0: jdbc:hive2://hadoop02:10000>
```

另外还有一种方式也可以去连接：先执行 beeline

然后按图所示输入：**!connect jdbc:hive2://hadoop02:10000** 按回车，然后输入用户名，这个用户名就是安装 hadoop 集群的用户名

```
hadoop@hadoop02 ~]$ beeline
Last login: Wed Jan 24 19:23:51 2018 from 192.168.123.1
hadoop@hadoop02 ~]$ beeline
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/hadoop/apps/apache-hive-2.3.2-bin/lib/log4j-slf4]
SLF4J: Found binding in [jar:file:/home/hadoop/apps/hadoop-2.7.5/share/hadoop/common/lib]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Beeline version 2.3.2 by Apache Hive
beeline> !connect jdbc:hive2://hadoop02:10000
Connecting to jdbc:hive2://hadoop02:10000
Enter username for jdbc:hive2://hadoop02:10000: hadoop
Enter password for jdbc:hive2://hadoop02:10000: *****
Connected to: Apache Hive (version 2.3.2)
Driver: Hive JDBC (version 2.3.2)
Transaction isolation: TRANSACTION_REPEATABLE_READ
0: jdbc:hive2://hadoop02:10000>
```

接下来便可以做 hive 操作

2.2.3、Web UI

1、下载对应版本的 src 包：apache-hive-2.3.2-src.tar.gz

2、上传，解压

tar -zxvf apache-hive-2.3.2-src.tar.gz

3、然后进入目录\${HIVE_SRC_HOME}/hwi/web，执行打包命令：

jar -cvf hive-hwi-2.3.2.war *

在当前目录会生成一个 hive-hwi-2.3.2.war

4、得到 hive-hwi-2.3.2.war 文件，复制到 hive 下的 lib 目录中

cp hive-hwi-2.3.2.war \${HIVE_HOME}/lib/

5、修改配置文件 hive-site.xml

```
<property>
    <name>hive.hwi.listen.host</name>
    <value>0.0.0.0</value>
    <description>监听的地址</description>
</property>
<property>
    <name>hive.hwi.listen.port</name>
    <value>9999</value>
    <description>监听的端口号</description>
</property>
<property>
    <name>hive.hwi.war.file</name>
    <value>lib/hive-hwi-2.3.2.war</value>
    <description>war 包所在的地址</description>
</property>
```

6、复制所需 jar 包

1、**cp \${JAVA_HOME}/lib/tools.jar \${HIVE_HOME}/lib**

2、再寻找三个 jar 包，都放入\${HIVE_HOME}/lib 目录：

commons-el-1.0.jar

jasper-compiler-5.5.23.jar

jasper-runtime-5.5.23.jar

不然启动 hwi 服务的时候会报错。

7、安装 ant

1、上传 ant 包：apache-ant-1.9.4-bin.tar.gz

2、解压

```
tar -zxvf apache-ant-1.9.4-bin.tar.gz -C ~/apps/
```

3、配置环境变量

```
vi /etc/profile
```

在最后增加两行：

```
export ANT_HOME=/home/hadoop/apps/apache-ant-1.9.4
```

```
export PATH=$PATH:$ANT_HOME/bin
```

配置完环境变量别忘记执行：source /etc/profile

4、验证是否安装成功

```
[root@hadoop02 lib]# ant -version
Apache Ant(TM) version 1.9.4 compiled on April 29 2014
```

8、上面的步骤都配置完，基本就大功告成了。进入\${HIVE_HOME}/bin 目录：

```
${HIVE_HOME}/bin/hive --service hwi
```

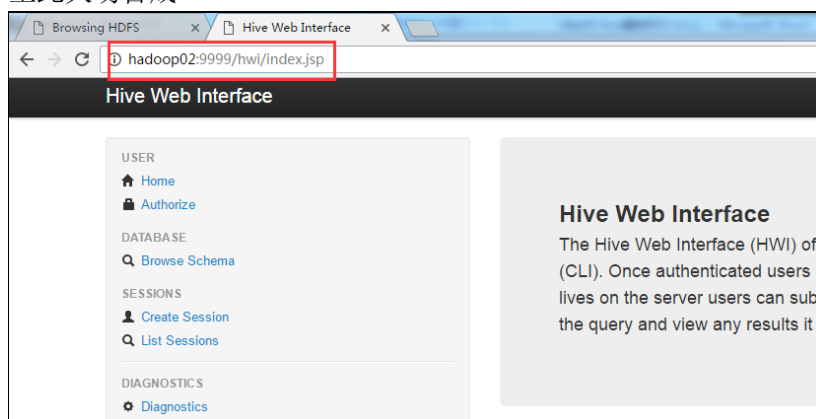
或者让在后台运行：

```
nohup bin/hive --service hwi > /dev/null 2> /dev/null &
```

9、前面配置了端口号为 9999，所以这里直接在浏览器中输入：

```
hadoop02:9999/hwi
```

10、至此大功告成



3、Hive 基本使用

1、创建库：create database if not exists mydb;

2、查看库：show databases;

3、切换数据库：use mydb;

4、创建表：create table if not exists t_user(id string, name string);

或 create table t_user(id string, name string) row format delimited fields terminated by ',';

5、查看表列表：show tables;

6、查看表的详细信息：desc formatted t_user;

7、插入数据：insert into table t_user values ('1','huangbo'), ('2','xuzheng'), ('3','wangbaoqiang');

8、查询数据：select * from t_user;

9、导入数据：

- a) 导入 HDFS 数据：load data inpath '/user.txt' into table t_user;
 - b) 导入本地数据：load data local inpath '/home/hadoop/user.txt' into table t_user;
- user.txt 的数据为：

4,liudehua
5,wuyanzu
6,liangchaowei

10、再次查询数据：select * from t_user;

小技能补充：

- 1、进入到用户的主目录，使用命令 **cat /home/hadoop/.hivehistory** 可以查看到 hive 执行的历史命令
- 2、执行查询时若想显示表头信息时，请执行命令：
Hive> **set hive.cli.print.header=true;**
- 3、hive 的执行日志的存储目录在\${java.io.tmpdir}/\${user.name}/hive.log 中，假如使用 hadoop 用户操作的 hive，那么日志文件的存储路径为：/temp/hadoop/hive.log