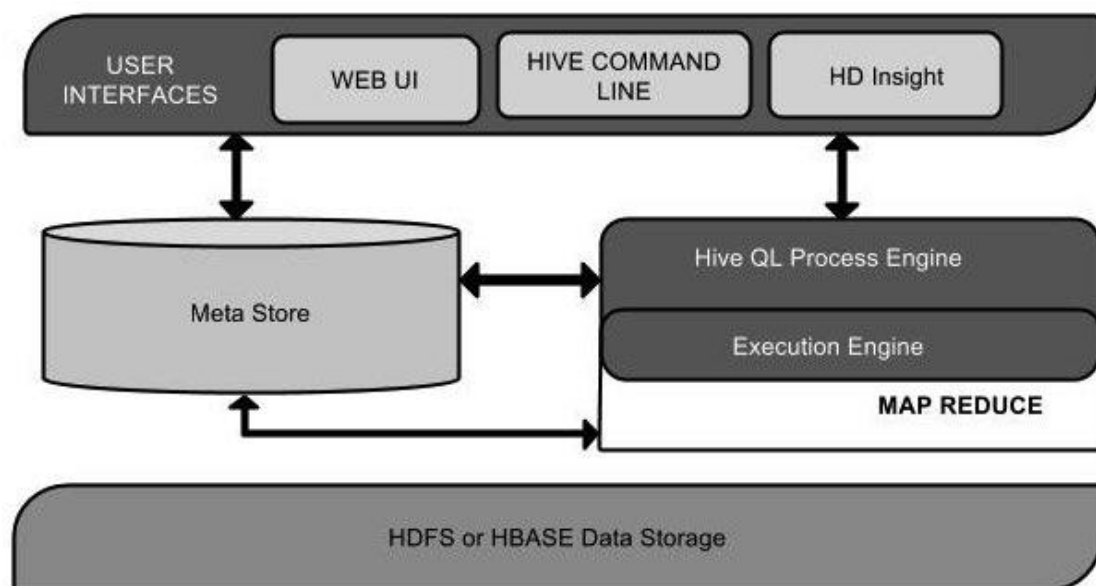


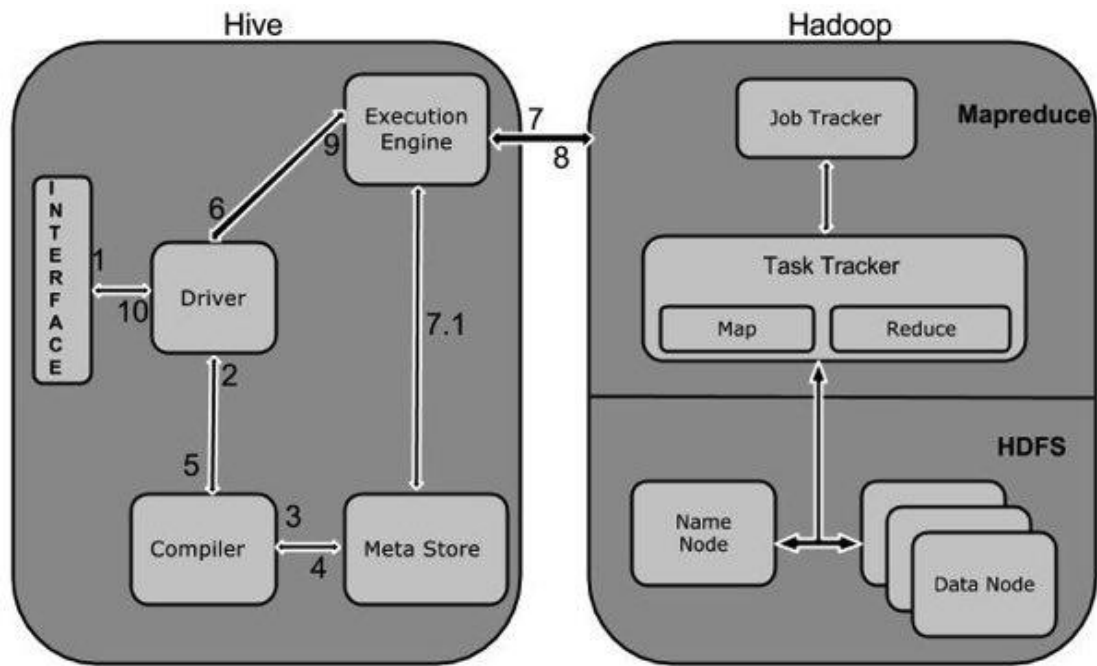
Hive 安装

什么是 Hive ?

Hive 是一个基于 Hadoop 文件系统之上的数据仓库架构。它为数据仓库的管理提供了许多功能：数据 ETL（抽取、转换和加载）工具、数据存储管理和大型数据集的查询和分析能力。同时 Hive 还定义了类 SQL 的语言 -- Hive QL. Hive QL 允许用户进行和 SQL 相似的操作，它可以将结构化的数据文件映射为一张数据库表，并提供简单的 SQL 查询功能。还允许开发人员方便地使用 Mapper 和 Reducer 操作，可以将 SQL 语句转换为 MapReduce 任务运行，这对 MapReduce 框架来说是一个强有力的支持。

Hive 架构及 workflow





从上面两张图中可以看出，Hive 的组成大致包括：

- 用户接口，包括 CLI, JDBC/ODBC, WebUI
- 元数据存储，通常是存储在关系数据库如 MySQL, Derby 中
- 解释器、编译器、优化器、执行器
- Hadoop, 用 HDFS 进行存储，利用 MapReduce 进行计算

Hive 与关系数据库的区别

Hive 在很多方面与传统关系数据库类似（例如支持 SQL 接口），但是其底层对 HDFS 和 MapReduce 的依赖意味着它的体系结构有别于传统关系数据库，而这些区别又影响着 Hive 所支持的特性，进而影响着 Hive 的使用。

我们可以列举一些简单区别：

- Hive 和关系数据库存储文件的系统不同，Hive 使用的是 Hadoop 的 HDFS（Hadoop 的分布式文件系统），关系数据库则是服务器本地的文件系统；
- Hive 使用的计算模型是 MapReduce，而关系数据库则是自己设计的计算模型；
- 关系数据库都是为实时查询的业务进行设计的，而 Hive 则是为海量数据做数据挖掘设计的，实时性很差；实时性的区别导致 Hive 的应用场景和关系数据库有很大的不同；
- Hive 很容易扩展自己的存储能力和计算能力，这个是继承 Hadoop 的，而关系数据库在这个方面要差很多。

Hive 应用场景

通过对 Hive 与传统关系数据库的比较之后，其实我们不难得出 Hive 可以应用于哪些场景。Hive 构建在基于静态批处理的 Hadoop 之上，Hadoop 通常都有较高的延迟并且在作业提交和调度的时候需要大量的开销。因此，Hive 不适合在大规模数据集上实现低延迟快速的查询。

Hive 并不适合那些需要低延迟的应用，例如，联机事务处理（OLTP）。Hive 查询操作过程严格遵守 Hadoop MapReduce 的作业执行模型，Hive 将用户的 HiveQL 语句通过解释器转换为 MapReduce 作业提交到 Hadoop 集群上，Hadoop 监控作业执行过程，然后返回作业执行结果给用户。Hive 并非为联机事务处理而设计，Hive 并不提供实时的查询和基于行级的数据更新操作。

Hive 的最佳使用场合是大数据集的批处理作业，例如，网络日志分析。

Hive 的数据存储

Hive 的存储是建立在 Hadoop 文件系统之上的。Hive 本身没有专门的数据存储格式，也不能为数据建立索引，因此用户可以非常自由地组织 Hive 中的表，只需要在创建表的时候告诉 Hive 数据中的列分隔符就可以解析数据了。

Hive 中主要包括 4 种数据模型：表（Table）、外部表（External Table）、分区（Partition）以及 桶（Bucket）。

Hive 的表和数据库中的表在概念上没有什么本质区别，在 Hive 中每个表都有一个对应的存储目录。而外部表指向已经在 HDFS 中存在的数据，也可以创建分区。Hive 中的每个分区都对应数据库中相应分区列的一个索引，但是其对分区的组织方式和传统关系数据库不同。桶在指定列进行 Hash 计算时，会根据哈希值切分数据，使每个桶对应一个文件。

Hive 安装与配置

一、Hive 运行模式

与 Hadoop 类似，Hive 也有 3 种运行模式：

- **内嵌模式**

将元数据保存在本地内嵌的 Derby 数据库中，这是使用 Hive 最简单的方式。但是这种方式缺点也比较明显，因为一个内嵌的 Derby 数据库每次只能访问一个数据文件，这就意味着它不支持多会话连接。

- **本地模式**

这种模式是将元数据保存在本地独立的数据库中（一般是 MySQL），这用就可以支持多会话和多用户连接了。

- **远程模式**

此模式应用于 Hive 客户端较多的情况。把 MySQL 数据库独立出来，将元数据保存在远端独立的 MySQL 服务中，避免了在每个客户端都安装 MySQL 服务而造成冗余浪费的情况。

二、下载安装 Hive

上节课我们已经了解到，Hive 是基于 Hadoop 文件系统之上的数据仓库。因此，安装 Hive 之前必须确保 Hadoop 已经成功安装。本次实验，使用 Hive V2.0.1 版本。Hive V2.0.1 可以在 Hadoop V2.4.x 以上环境中工作。

- 登陆 hadoop 账户

```
su - hadoop
```

- 下载 hive

```
wget http://archive.apache.org/dist/hive/hive-2.0.1/apache-hive-2.0.1-bin.tar.gz
```

- 对 apache-hive-2.0.1-bin.tar.gz 进行解压：

```
tar xzvf apache-hive-2.0.1-bin.tar.gz
```

- 修改解压后的目录为 hive

```
mv apache-hive-2.0.1-bin hive
```

三、配置系统环境变量

退出 hadoop 账户到 root 账户

```
exit
```

修改/etc/profile 文件，这个我们在 Hadoop 和 HBase 的课程 中也修改过，应该比较熟悉了。

```
sudo vi /etc/profile
```

配置以下内容到 profile 中去，并保存

```
export HIVE_HOME=/hadoop/hive
```

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

修改完成后，让修改生效：

```
source /etc/profile
```

四、内嵌模式配置

(1)hive-site.xml

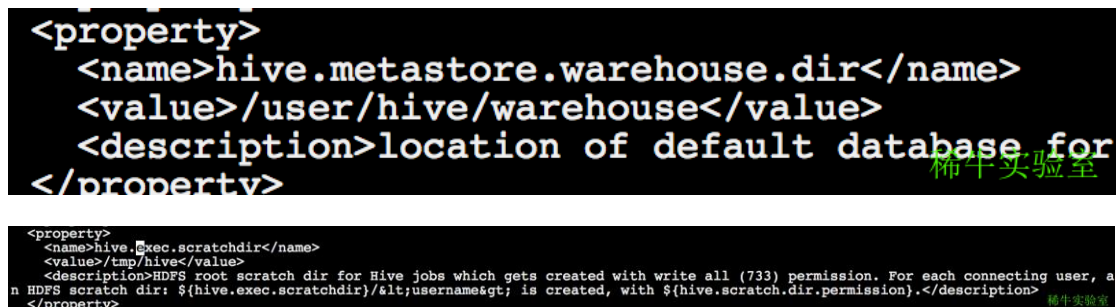
首先切回 Hadoop 用户

```
su - hadoop
```

\$HIVE_HOME/conf 对应的是 Hive 的配置文件路径，类似于之前学习的 HBase，该路径下的 hive-site.xml 是 Hive 工程的配置文件。默认情况下，该文件并不存在，我们需要拷贝它的模版来实现（这里暂时不需要修改，先拷贝）：

```
cp hive/conf/hive-default.xml.template hive/conf/hive-site.xml
```

hive-site.xml 的主要配置有一下，不需要修改：



```
<property>
  <name>hive.metastore.warehouse.dir</name>
  <value>/user/hive/warehouse</value>
  <description>location of default database for
</property>

<property>
  <name>hive.exec.scratchdir</name>
  <value>/tmp/hive</value>
  <description>HDFS root scratch dir for Hive jobs which gets created with write all (733) permission. For each connecting user, a
n HDFS scratch dir: ${hive.exec.scratchdir}/<username> is created, with ${hive.scratch.dir.permission}.</description>
</property>
```

- hive.metastore.warehouse.dir 该参数指定了 Hive 的数据存储目录，默认位置在 HDFS 上面的 /user/hive/warehouse 路径下。
- hive.exec.scratchdir 该参数指定了 Hive 的数据临时文件目录，默认位置为 HDFS 上面的 /tmp/hive 路径下。

同时我们还要修改 Hive 目录下 /conf/hive-env.sh 文件（请根据自己的实际路径修改），该文件默认也不存在，同样是拷贝它的模版来修改：

```
cp hive/conf/hive-env.sh.template hive/conf/hive-env.sh
```

然后修改 hive/conf/hive-env.sh

```
vi hive/conf/hive-env.sh
```

注意，以下配置需要在该文件的 vim 编辑模式，通过“/”查找命令，先找到对应的 key，再打开注释，修改 value。

```
export HADOOP_HEAPSIZE=100
```

```
# Set HADOOP_HOME to point to a specific hadoop install directory
```

```
HADOOP_HOME=/hadoop/hadoop
```

```
# Hive Configuration Directory can be controlled by:
```

```
export HIVE_CONF_DIR=/hadoop/hive/conf
```

```
# Folder containing extra libraries required for hive compilation/execution can be controlled by:
```

```
export HIVE_AUX_JARS_PATH=/hadoop/hive/lib
```

(2)创建必要目录

前面我们看到 `hive-site.xml` 文件中有两个重要的路径，我们需要先启动 Hadoop，确认是否有 `/user` 和 `/tmp` 这两个 `hdfs` 文件夹：

```
start-all.sh
```

```
hadoop dfs -ls /
```

如果没有发现上面提到的路径，需要自己新建这些目录，并且给它们赋予用户写（W）权限。

```
hadoop dfs -mkdir /user
```

```
hadoop dfs -mkdir /user/hive
```

```
hadoop dfs -mkdir /user/hive/warehouse
```

```
hadoop dfs -mkdir /tmp/hive
```

```
hadoop dfs -chmod 777 /user/hive/warehouse
```

```
hadoop dfs -chmod 777 /tmp/hive
```

如果你遇到 `no such file or directory` 类似的错误，就一步一步新建目录，例如：

```
hadoop dfs -mkdir /tmp
```

```
hadoop dfs -mkdir /tmp/hive
```

检查是否新建成功 `hadoop dfs -ls /` 以及 `hadoop dfs -ls /user/hive/` :

```
$ hadoop dfs -ls /user/hive/
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

17/03/31 09:04:50 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Found 1 items
drwxrwxrwx - hadoop supergroup          0 2017-03-31 09:02 /user/hive/warehouse
```

(3)修改 io.tmpdir 路径

同时，要修改 `hive-site.xml` 中所有包含 `${system:java.io.tmpdir}` 字段的 `value` 即路径，你可以自己新建一个目录来替换它，例如 `/hadoop/hive/iotmp`。同样注意修改写权限。

注 1: `vi` 下命令模式直接按 `/` 表示搜索，后面跟你的关键词，比如搜索 `hello`，则为 `/hello`，再回车即可，如接着按 `n` 可以查找下一个

注 2: `vi` 下使用 “`%s` 可以做替换”。例如，键入冒号，再输入 `%s/${system:java.io.tmpdir}/hadoop/hive/gc`，回车，根据提示连续按 `y` 确认替换，可以将所有 “`${system:java.io.tmpdir}`” 替换为 “`/hadoop/hive`”。详见 <http://blog.csdn.net/ailiwanzi/article/details/44859983>

以下是一个修改示例：

将

```
<value>${system:java.io.tmpdir}/${system:user.name}</value>
```

改成

```
<value>/hadoop/hive/iotmp</value>
```

五、本地模式配置

我们替换默认的 `Derby` 数据库为 `MySQL` 数据库。

(1)下载安装 MySQL

先进入到 `root` 用户

```
sudo apt-get update
```

```
sudo apt-get install mysql-server
```

安装过程中，设置 root 密码为 root。 然后启动它：

```
sudo service mysql start
```

用 root 用户登录

```
mysql -uroot -p
```

```
$ mysql -uroot -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 43
Server version: 5.5.54-0ubuntu0.14.04.1 (Ubuntu)

Copyright (c) 2000, 2016, 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> 
```

犀牛实验室

创建 hive 数据库：

```
create database hive;
```

```
grant all on hive.* to 'hive'@'localhost' identified by 'hive';
```

```
flush privileges;
```

```
mysql> create database hive;
Query OK, 1 row affected (0.00 sec)

mysql> grant all on hive.* to 'hive'@'localhost' identified by 'hive';
Query OK, 0 rows affected (0.00 sec)

mysql> flush privileges;
Query OK, 0 rows affected (0.00 sec)
```

犀牛实验室

我们还需要下载一个 MySQL 的 JDBC 驱动包。这里使用的是 mysql-connector-java-5.1.35-bin.jar，你需要将其复制到 \$HIVE_HOME/lib 目录下：

```
su - hadoop
```

```
wget https://dev.mysql.com/get/Downloads/Connector-J/mysql-connector-java-5.1.41.tar.gz
```

```
tar -xzf mysql-connector-java-5.1.41.tar.gz
```

```
cp mysql-connector-java-5.1.41/mysql-connector-java-5.1.41-bin.jar $HIVE_HOME/lib
```


(2)修改 hive-site.xml 配置文件

最后，依然是修改 \$HIVE_HOME/conf 下的 hive-site.xml 文件，把默认的 Derby 修改为 MySQL：

```
<property>
  <name>javax.jdo.option.ConnectionURL</name>
  //所连接的 MySQL 数据库实例
  <value>jdbc:mysql://localhost:3306/hive?createDatabase
IfNotExist=true</value>
</property>

<property>
  <name>javax.jdo.option.ConnectionDriverName</name>
  //连接的 MySQL 数据库驱动
  <value>com.mysql.jdbc.Driver</value>
</property>

<property>
  <name>javax.jdo.option.ConnectionUserName</name>
  //连接的 MySQL 数据库用户名
  <value>hive</value>
</property>

<property>
  <name>javax.jdo.option.ConnectionPassword</name>
  //连接的 MySQL 数据库密码
  <value>hive</value>
</property>
```

初始化 schema

```
schematool -dbType mysql -initSchema
```

```
$ schematool -dbType mysql -initSchema
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/hadoop/hive/lib/log4j-slf4j-impl-2.4.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/hadoop/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Metastore connection URL:      jdbc:mysql://localhost:3306/hive?createDatabaseIfNotExist=true
Metastore Connection Driver :  com.mysql.jdbc.Driver
Metastore connection User:    hive
Starting metastore schema initialization to 2.0.0
Initialization script hive-schema-2.0.0.mysql.sql
Initialization script completed
schemaTool completed
```

犀牛实验室

(3)启动 hive

```
$ hive
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/hadoop/hive/lib/log4j-slf4j-impl-2.4.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/hadoop/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
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:/hadoop/hive/lib/hive-common-2.0.1.jar!/hive-log4j2.properties
Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e . tez, spark) or using Hive 1.X releases.
hive>
```

绿牛实验室