Hive日志调试

写在前面的话
，学Hive这么久了，发现目前国内还没有一本完整的介绍Hive的书籍，而且互联网上面的资料很乱，于是我决定写一些关于《Hive的那些事》
序列文章，分享给大家。我会在接下来的时间整理有关Hive的资料，如果对Hive的东西感兴趣，请关注本博客。https://www.iteblog.com/archives/tag/hive-technology/

这些天看到很多人在使用Hive的过程遇到这样或那样的错误，看着那些少的可怜的错误日志出错，一直找不到原因。后来我给他们介绍了修改日志输出级别之后，错误原因很快得到定位。于是乎我写了这篇博文。希望那些在使用HQL的过程中遇到问题，通过这里介绍的方法进行调试而定位到错误，从而少走弯路。好了，废话不多说进入正文。

在很多程序中，我们都可以通过输出日志的形式来得到程序的运行情况，通过这些输出日志来调试程序，Hive也不例外。

在Hive中，使用的是Log4j来输出日志，默认情况下，CLI是不能将日志信息输出到控制台的。在Hive0.13.0之前版本，默认的日志级别是WARN，从Hive0.13.0开始，默认的日志级别是INFO。默认的日志存放在/tmp/<user.name>文件夹的hive.log文件中，全路径就是/tmp/<user.name>/hive.log。

需要注意的一个bug
：在本地模式情况下，log文件名为".log"，而不是"hive.log"，可以在这里看到https://issues.apache.org/jira/browse/HIVE-5528，这个bug将会在Hive0.13.0中得到解决。

在默认的日志级别情况下，是不能将DEBUG信息输出，这样一来出现的各种详细的错误信息都是不能数错的。但是我们可以通过以下两种方式修改log4j输出的日志级别，从而利用这些调试日志进行错误定位，具体做法如下：

```
[wyp@master ~]$ hive --hiveconf hive.root.logger=DEBUG,console
```

或者在${HIVE_HOME}/conf/hive-log4j.properties文件中找到hive.root.logger属性，并将其修改为下面的设置

```
hive.root.logger=DEBUG,console
```

上面两种方法的设置各有优劣，方法一的设定只是对本次会话有效，下次如果还想修改日志输出级别需要重新设定，但是不是每时每刻都需要修改日志的输出级别，所以在有时需要修改输出的日志级别，有时不需要的时候可以用这种方法；方法二将日志输出级别设定到文件中去了，

这个设定是对所有的用户都生效，而且每次使用HQL的时候都会输出一大堆的日志，这种情况适合那些无时无刻都需要HQL的运行日志的用户。

不过，我们可以设定自己的log4j配置文件，通过修改-hiveconf hive.log4j.file=/home/iteblog/hive-log4j.properties参数指定。这样设置对别人不会产生影响。

在《Hive几种参数配置方法》文章中我们介绍了Hive三种参数配置方法，其中就提到了某些系统级的参数，在HQL中设定是无效的。这里就是一个很好的例子。因为设定log的参数读取在会话建立以前已经完成了。这也就说，我们不能通过下面的方法来修改log4j的日志输出级别：

```
hive> set hiveconf:hive.root.logger=DEBUG,console;
```

这样你在进入CLI的时候将会得到一些类似下面的调试信息：

```
[wyp@master /home/q/hive-0.11.0-bin/conf]$ hive
.....为了篇幅，省略了很多.....
13/12/25 15:14:54 DEBUG parse.VariableSubstitution: Substitution is on: hive
.....为了篇幅，省略了很多.....
13/12/25 15:14:54 DEBUG security.Groups:  Creating new Groups object
13/12/25 15:14:54 DEBUG util.NativeCodeLoader: Trying to load the c...
library for your platform... using builtin-java classes where applicable
13/12/25 15:14:54 DEBUG security.JniBasedUnixGroupsMappingWithFallback:
Falling back to shell based
13/12/25 15:14:54 DEBUG security.JniBasedUnixGroupsMappingWithFallback:
Group mapping impl=org.apache.hadoop.security.ShellBasedUnixGroupsMapping
13/12/25 15:14:54 DEBUG security.UserGroupInformation: hadoop login
13/12/25 15:14:54 DEBUG security.UserGroupInformation: using local
 user:UnixPrincipal: wyp
13/12/25 15:14:54 DEBUG security.UserGroupInformation:
UGI loginUser:wyp (auth:SIMPLE)
.....为了篇幅，省略了很多.....
```

下面举个日志调试的例子，在没有修改日志输出级别之前，有下面的查询所有表的HQL如下：

```
hive> show tables;
FAILED: Error in metadata: java.lang.RuntimeException: Unable to instantiate
        org.apache.hadoop.hive.metastore.HiveMetaStoreClient
```

```
FAILED: Execution Error, return code 1 from
                org.apache.hadoop.hive.ql.exec.DDLTask
hive>
```

得到上面的错误，我们从上面的错误输出只知道是元数据有问题。具体的错误也不知道，这时候如果我们修改了日志调试级别hive.root.logger=DEBUG,console，我们再来看看运行上面语句的错误输出：

```
hive> show tables;
.....为了篇幅，省略了很多.....
13/12/25 15:23:58 INFO metastore.ObjectStore: ObjectStore, initialize called
13/12/25 15:23:58 ERROR Datastore.Schema: Failed initialising database.
Access denied for user 'datalog5'@'l-datalog5.data.cn1' (using password: YES)
org.datanucleus.exceptions.NucleusDataStoreException:
Access denied for user 'datalog5'@'l-datalog5.data.cn1' (using password: YES)
    at org.datanucleus.store.rdbms.ConnectionFactoryImpl
       $ManagedConnectionImpl.getConnection(ConnectionFactoryImpl.java:536)
    at org.datanucleus.store.rdbms.RDBMSStoreManager.<init>
                                (RDBMSStoreManager.java:290)
    at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method)
.....为了篇幅，省略了很多.....
```

通过上面的错误堆栈我们就可以将问题定位到是连接数据库出现了问题，这么一来错误的定位范围就大大减少了，我们可以查看书Hive-site.xml文件中连接数据库的配置是否正确等来解决上述问题。