三、Hive 数据仓库安装部署及测试

安装 Mysql

sudo apt-get install mysql-server

安装完成后设置数据库

sudo mysql_secure_installation

root 用户的密码，123456

配置 Mysql

Sudo mysql –u root –p
使用初始化时设置的 root 密码登录

新增 hive 用户，并给权限：

create user 'hive' identified by 'hive';
grant all privileges on *.* to 'hive' with grant option;
flush privileges;

Hive 安装与配置

安装 hive

将 apache-hive-3.1.2-bin.tar.gz 解压在/usr/local 目录下

sudo tar –zxvf apache-hive-3.1.2-bin.tar.gz –C /usr/local

重命名文件夹为 hive 文件夹，并修改权限

mv /usr/local/ apache-hive-3.1.2-bin /usr/local/hive
sudo chown -R hadoop:hadoop /usr/local/hive

把 mysql 的 jdbc 的驱动 mysql-connector-java-8.0.18-bin.jar 拷贝到
\usr\local\hive\lib 目录下

cp mysql-connector-java-8.0.18.jar /usr/local/hive/lib

Hive 的配置

在 /usr/local/hive/conf 目录下创建 hive-site.xml 文件:

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

在 server 端配置 hive-site.xml

ConnectionURL 属性用于设置 mysql 服务所在地址与端口，这里 mysql-server 在本地，hive.metastore.warehouse.dir 是在 HDFS 上的文件路径，hive.metastore.local 的值为 true 表示对 metastore 的访问为本地模式。

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

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

<value>jdbc:mysql://localhost:3306/db_hive?createDatabaseIfNotExist=true</value>
    </property>

    <property>
        <name>javax.jdo.option.ConnectionDriverName</name>
        <value>com.mysql.jdbc.Driver</value>
    </property>

    <property>
        <name>javax.jdo.option.ConnectionUserName</name>
        <value>hive</value>
    </property>

    <property>
        <name>javax.jdo.option.ConnectionPassword</name>
        <value>hive</value>
    </property>

    <property>
        <name>hive.metastore.warehouse.dir</name>
        <value>/hive/warehouse</value>
    </property>
```

```
    <property>
        <name>hive.metastore.local</name>
        <value>true</value>
    </property>


    ////////////////////////////////////////////////////////////////
    <property>
        <name>yarn.resourcemanager.hostname</name>
        <value>192.168.150.131</value>
    </property>

    <property>
        <name>yarn.resourcemanager.address</name>
        <value>master:8032</value>
    </property>

    <property>
        <name>yarn.resourcemanager.schedulaer.address</name>
        <value>master:8030</value>
    </property>

    <property>
        <name>yarn.resourcemanager.resource-tracker.address</name>
        <value>master:8031</value>
    </property>


</configuration>
```

在 client 端配置 hive-site.xml

hive.metastore.uris 项指向提供数据库访问的 metastore 服务端，值须为 IP 地址。由于设置了 uris 的内容，因而对于 metastore 的访问默认为远程模式。

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<configuration>
    <property>
        <name>hive.metastore.warehouse.dir</name>
        <value>/hive/warehouse</value>
```

```
    </property>

    <property>
        <name>hive.metastore.local</name>
        <value>false</value>
    </property>

    <property>
        <name>hive.metastore.uris</name>
        <value>thrift://192.168.150.131:9083</value>
    </property>

    <property>
        <name>yarn.resourcemanager.hostname</name>
        <value>192.168.150.131</value>
    </property>

    <property>
        <name>yarn.resourcemanager.address</name>
        <value>master:8032</value>
    </property>

    <property>
        <name>yarn.resourcemanager.schedulaer.address</name>
        <value>master:8030</value>
    </property>

    <property>
        <name>yarn.resourcemanager.resource-tracker.address</name>
        <value>master:8031</value>
    </property>
</configuration>
```

运行 Hive

检查 jline 和 guava 版本，hive 与 hadoop 的 jline 和 guava 版本不对应可能导致运行错误，先删除之前的旧版本，然后将较新的 jar 包拷贝至旧的一方的对应目录下：
以 jline 为例：

```
cp /usr/local/hive/lib/jline-2.12.jar /usr/local/hadoop/share/hadoop/yarn/lib


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

cp guava-27.0-jre.jar /usr/local/hive/lib/
```

接下来的步骤只需在 master 上进行:

启动 hadoop：/usr/local/hadoop/sbin/start-all.sh

初始化 Schema
schematool –dbType mysql –initSchema

服务端启动 metastore 服务

hive --service metastore

接着在 slave1 和 slave2 启动 hive 的客户端

hive

四、spark 的配置

安装 scala 2.11.12（可选）

Sudo apt install scala

若版本不是 2.11.12 的话推荐去官网下载安装包手动安装
然后追加环境变量如下:
export SCALA_HOME=/usr/share/scala-2.11
export PATH=$PATH:$SCALA_HOME/bin

编译 Spark

1.下载 spark2.3.0 源码
2.解压
3.进入 spark2.3.0 目录，编译

①
./build/mvn -Pyarn -Phadoop-2.7 -Dhadoop.version=3.2.0 -DskipTests clean package

②进入 dev/
./make-distribution.sh --name 3.2.0-nohive --tgz -Pyarn -Phadoop-2.7 -Dhadoop.version=3.2.0

这里的几个参数:
–name 3.2.0-nohive 是编译文件的名字参数
-Pyarn 是支持 yarn

-Phadoop-2.7 是支持的 hadoop 版本，一开始使用的是 3.2 后来提示 hadoop3.2 不存在，只好改成 2.7，编译成功
-Dhadoop.version=3.2.0 运行环境

配置 Spark

解压文件到/usr/local 下，重命名文件夹并修改属主
sudo tar -xzvf spark-2.3.0-bin-3.2.0-nohive.tgz -C /usr/local/
sudo mv /usr/local/spark-2.3.0-bin-3.2.0-nohive /usr/local/spark
sudo chown -R hadoop:hadoop /usr/local/spark/

利用 spark 的 template 文件生成配置文件
cp /usr/local/spark/conf/spark-env.sh.template /usr/local/spark/conf/spark-env.sh

cp /usr/local/spark/conf/slaves.template /usr/local/spark/conf/slaves

cp /usr/local/spark/conf/spark-defaults.conf.template /usr/local/spark/conf/spark-defaults.conf

修改 spark-env.sh，在文件末尾添加如下内容
export HADOOP_HOME=/usr/local/hadoop
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
export HADOOP_CONF_DIR=${HADOOP_HOME}/etc/hadoop
export SPARK_MASTER_IP=master
export SPARK_LOCAL_DIRS=/usr/local/spark
export SPARK_WORKER_MEMORY=512M
export SPARK_EXECUTOR_MEMORY=512M
export SPARK_DRIVER_MEMORY=512M
export SPARK_EXECUTOR_CORES=1

这一步是为了配置 spark 的运行参数，hadoop_conf_dir 的设置是为了让 spark 运行在 yarn 上。几个 memory 命令分别用于设置 driver 和 executor 进程的内存，executor_cores 设置的是每个 executor 进程的 CPU cores 的数量，这些设置请依据自己的电脑实际可负载情况设置。

修改 slaves 文件，在文件末尾添加其他节点 IP
vi /usr/local/spark/conf/slaves

修改成
master
slave1
slave2

修改 spark-defaults.conf，在文件末尾添加如下内容：

vi /usr/local/spark/conf/spark-defaults.conf
spark.executor.extraJavaOptions -XX:+PrintGCDetails -Dkey=value -Dnumbers="one two three"
spark.eventLog.enabled true
spark.eventLog.dir hdfs://master:9000/historyserverforSpark
spark.yarn.historyServer.address master:18080
spark.history.fs.logDirectory hdfs://master:9000/historyserverforSpark
spark.speculation true

这一步是为保存 spark 的运行日志，并且是保存到 hdfs 上的文件夹里面，方便运维。
将配置好的 spark 文件夹传到 slave1、slave2。

确认 hadoop 中的 yarn-site.xml 文件存在以下属性：
<configuration>

<!-- Site specific YARN configuration properties -->
<property>
<name>yarn.nodemanager.aux-services</name>
<value>mapreduce_shuffle</value>
</property>
<property>
<name>yarn.log-aggregation-enable</name>
<value>true</value>
</property>
</configuration>
这一步是为了开启日志整合功能。

运行 spark

开启 hadoop 集群
/usr/local/hadoop/sbin/start-all.sh
在 spark 中创建 historyserverforSpark 文件夹
/usr/local/hadoop/bin/hdfs dfs -mkdir /historyserverforSpark

运行 spark
/usr/local/spark/sbin/start-all.sh

可以进入 spark 的 webui 查看是否成功启动：192.168.150.131:8080/
可以进入 spark 的 webui 查看节点是否成功启动：192.168.150.131:8080/cluster

后续 Hive 对接

拷贝 spark-defaults.conf 到{hive-home}/conf 目录下

拷贝 spark/jars 的三个包到{hive-home}/lib:

scala-library
spark-core
spark-network-common

在 hive-site.xml 中进行全局设置

```
<property>
    <name>hive.execution.engine</name>
    <value>spark</value>
</property>
<property>
    <name>hive.enable.spark.execution.engine</name>
    <value>true</value>
</property>
<property>
    <name>spark.master</name>
    <value>spark://master:7077</value>
</property>
```

验证 hive on spark

```
create table tb_user(id int,name string ,age int);
insert into tb_user values(1,'name1',11),(2,'name2',12),(3,'name3',13);
```

参考链接:

1. https://www.jianshu.com/p/26c49450ba58
2. https://blog.csdn.net/weixx3/article/details/80782479
3. https://blog.csdn.net/CleverCode/article/details/50574695
4. https://blog.csdn.net/hongweigg/article/details/39995211
5. https://blog.csdn.net/qq_32635069/article/details/80859790