4 Hive

4.1 实验介绍

4.1.1 关于本实验

Hive 是重要的数据仓库工具，在数据挖掘、数据汇总、统计分析等领域有重要作用。

鲲鹏 BigData Pro 大数据解决方案使用鲲鹏服务器，其有 ARM 多核架构的特点。提升数据处理的并发能力，可以有效提升大数据处理效率，Hive 数据仓库组件收益明显。

4.1.2 实验目的

掌握鲲鹏 BigData Pro 解决方案的 Hive 部署方法。

掌握 Hive 的常用操作。

4.2 实验任务 1

4.2.1 启动 Hadoop 集群

步骤 1 在 node1 节点执行以下命令：

```
> start-dfs.sh ; start-yarn.sh
```

返回信息中有以下内容，表示 Hadoop 集群启动成功：

```
Starting namenodes on [node1]
Starting secondary namenodes [node1]
starting yarn daemons
```

此外，关闭 Hadoop 集群的命令为：

```
stop-dfs.sh && stop-yarn.sh
```

4.2.2 验证 Hadoop 状态

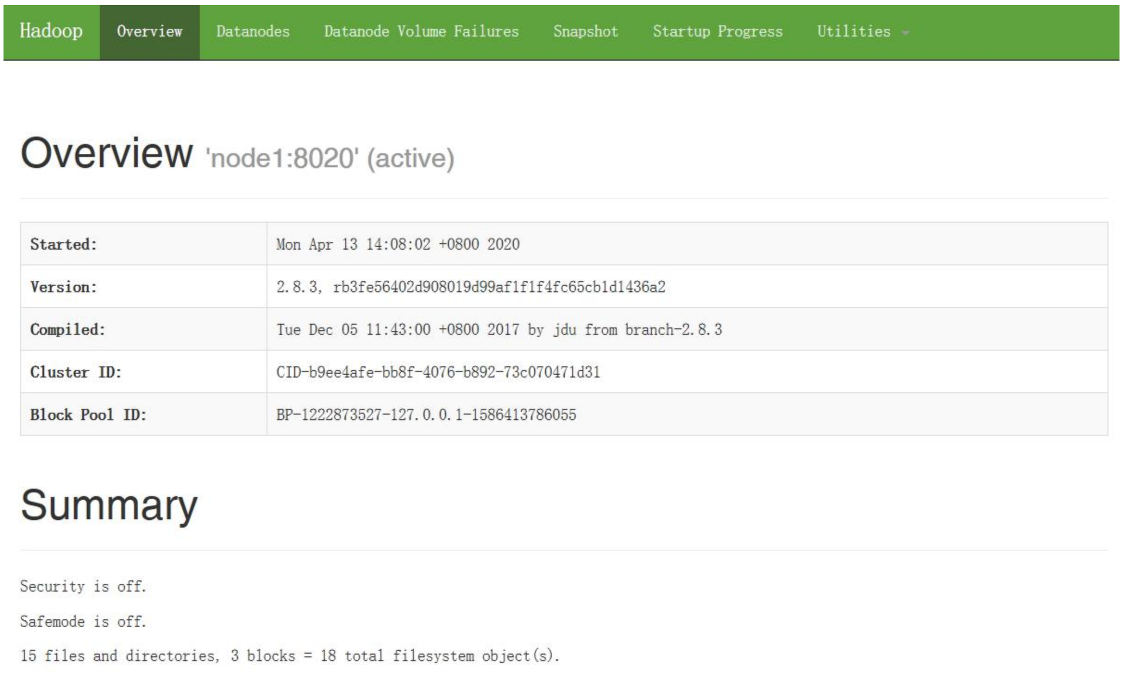
使用 jps 命令在 node1-4 中查看 Java 进程，node1 中有 NameNode、SecondaryNameNode 和 ResourceManager 进程：

```
[root@node1 ~]# jps
1538 WrapperSimpleApp
5732 SecondaryNameNode
5508 NameNode
6205 Jps
5918 ResourceManager
```

node2-4 中有 NodeManager 和 Datanode 进程：

```
[root@node2 ~]# jps
3026 Jps
2740 DataNode
1515 WrapperSimpleApp
2862 NodeManager
```

访问 http://node1 弹性公网 IP:50070，可以访问 HDFS 的 WebUI 界面：



Overview 'node1:8020' (active)

Started:	Mon Apr 13 14:08:02 +0800 2020
Version:	2.8.3, rb3fe56402d908019d99af1f1f4fc65cb1d1436a2
Compiled:	Tue Dec 05 11:43:00 +0800 2017 by jdu from branch-2.8.3
Cluster ID:	CID-b9ee4afe-bb8f-4076-b892-73c070471d31
Block Pool ID:	BP-1222873527-127.0.0.1-1586413786055

Summary

Security is off.

Safemode is off.

15 files and directories, 3 blocks = 18 total filesystem object(s).

图 4-1 HDFS 的 WebUI 界面

4.3 实验任务 2

本节主要介绍 BigData Pro 解决方案中 Hive 组件的部署方法，首先在 node1 上部署 Hive 组件，然后将组件包分发到 node2-4 节点上，最后统一启动 Hive。

4.3.1 在 node1 上安装 MySQL

步骤 1 添加 MySQL 用户组和用户

添加 MySQL 用户组和 MySQL 用户，用于隔离 MySQL 进程，在 node1 节点执行以下命令：

```
> groupadd -r mysql && useradd -r -g mysql -s /sbin/nologin -M mysql
```

步骤 2 安装 MySQL 依赖库

安装依赖库 libaio.aarch64 以及 libaio-devel.aarch64

```
[root@node1 ~]# yum install -y libaio*
...
Installed:
  libaio.aarch64 0:0.3.109-13.el7
libaio-devel.aarch64 0:0.3.109-13.el7


Complete!
# 表示安装完成
```

步骤 3 解压并重命名 MySQL 安装包

```
> tar -xvf /home/extend_tools/mysql-5.7.27-aarch64.tar.gz -C
/usr/local/
> mv /usr/local/mysql-5.7.27-aarch64 /usr/local/mysql
```

步骤 4 配置 MySQL

```
> mkdir -p /usr/local/mysql/logs
> chown -R mysql:mysql /usr/local/mysql
> ln -sf /usr/local/mysql/my.cnf /etc/my.cnf
> cp -rf /usr/local/mysql/extra/lib* /usr/lib64/
> mv /usr/lib64/libstdc++.so.6 /usr/lib64/libstdc++.so.6.old
> ln -s /usr/lib64/libstdc++.so.6.0.24 /usr/lib64/libstdc++.so.6
```

步骤 5 设置 MySQL 开机启动

```
> cp -rf /usr/local/mysql/support-files/mysql.server
/etc/init.d/mysqld
> chmod +x /etc/init.d/mysqld
> /sbin/chkconfig mysqld on
```

步骤 6 添加 MySQL 环境变量

执行命令：

```
vim /etc/profile
```

在末尾添加 MySQL 环境变量：

```
export MYSQL_HOME=/usr/local/mysql
export PATH=$PATH:$MYSQL_HOME/bin
```

然后执行下面命令，确保环境变量生效：

```
> source /etc/profile
```

步骤 7 初始化 MySQL

执行以下命令，初始化 MySQL：

```
> mysqld --initialize-insecure --user=mysql --basedir=/usr/local/mysql
--datadir=/usr/local/mysql/data
```

初始化完成后，启动 MySQL

```
> systemctl start mysqld
```

查看 MySQL 启动状态是否正常

```
> systemctl status mysqld
● mysqld.service - LSB: start and stop MySQL
   Loaded: loaded (/etc/rc.d/init.d/mysqld; bad; vendor preset:
disabled)
   Active: active (running) since Mon 2020-04-20 11:29:50 CST; 8s ago
     Docs: man:systemd-sysv-generator(8)
  Process: 3990 ExecStart=/etc/rc.d/init.d/mysqld start (code=exited,
status=0/SUCCESS)
   CGroup: /system.slice/mysqld.service
           ├─4007 /bin/sh /usr/local/mysql/bin/mysqld_safe
--datadir=/usr/local/mysql/data --pid-file=/usr/local/m...
           └─5572 /usr/local/mysql/bin/mysqld
--basedir=/usr/local/mysql --datadir=/usr/local/mysql/data
--plugin-...
```

步骤 8 MySQL 安全配置

执行如下命令，进行 MySQL 安全配置

```
> mysql_secure_installation

Securing the MySQL server deployment.

Connecting to MySQL using a blank password.

VALIDATE PASSWORD PLUGIN can be used to test passwords
and improve security. It checks the strength of password
and allows the users to set only those passwords which are
secure enough. Would you like to setup VALIDATE PASSWORD plugin?
```

```
### 是否需要安装密码强度检测插件
Press y|Y for Yes, any other key for No: y

There are three levels of password validation policy:

LOW    Length >= 8
MEDIUM Length >= 8, numeric, mixed case, and special characters
STRONG Length >= 8, numeric, mixed case, special characters and dictionary
file

### 输入 0 表示密码认证策略强度低，1 表示策略强度中，2 表示强度高
Please enter 0 = LOW, 1 = MEDIUM and 2 = STRONG: 0
Please set the password for root here.

### 设置 root 自定义用户密码， 设置完成后请做好个人记录
New password:
Re-enter new password:
### 显示自定义 root 密码评测结果
Estimated strength of the password: 100
### 是否继续使用上一步设置的密码，输入 y 继续使用
Do you wish to continue with the password provided?(Press y|Y for Yes,
any other key for No) : y
By default, a MySQL installation has an anonymous user,
allowing anyone to log into MySQL without having to have
a user account created for them. This is intended only for
testing, and to make the installation go a bit smoother.
You should remove them before moving into a production
environment.
### 是否移除匿名用户，输入 Y 移除
Remove anonymous users? (Press y|Y for Yes, any other key for No) : y
Success.


Normally, root should only be allowed to connect from
'localhost'. This ensures that someone cannot guess at
the root password from the network.
### 是否禁止 root 用户远程登录权限，输入 n 不禁止
Disallow root login remotely? (Press y|Y for Yes, any other key for No) :
n
```

```
 ... skipping.
By default, MySQL comes with a database named 'test' that
anyone can access. This is also intended only for testing,
and should be removed before moving into a production
environment.


### 是否删除 test 数据库，输入回车键，跳过选择
Remove test database and access to it? (Press y|Y for Yes, any other key
for No) :


 ... skipping.
Reloading the privilege tables will ensure that all changes
made so far will take effect immediately.
### 是否重新加载权限表，输入回车键，跳过选择
Reload privilege tables now? (Press y|Y for Yes, any other key for No) :


 ... skipping.
### 完成!
All done!
```

重启 MySQL 并查看 MySQL 状态

```
> systemctl restart mysqld
> systemctl status mysqld
● mysqld.service - LSB: start and stop MySQL
   Loaded: loaded (/etc/rc.d/init.d/mysqld; bad; vendor preset:
disabled)
   Active: active (running) since Mon 2020-04-20 11:44:19 CST; 2s ago
     Docs: man:systemd-sysv-generator(8)
  Process: 5651 ExecStop=/etc/rc.d/init.d/mysqld stop (code=exited,
status=0/SUCCESS)
  Process: 5685 ExecStart=/etc/rc.d/init.d/mysqld start (code=exited,
status=0/SUCCESS)
   CGroup: /system.slice/mysqld.service
           ├─5703 /bin/sh /usr/local/mysql/bin/mysqld_safe
--datadir=/usr/local/mysql/data --pid-file=/usr/local/m...
           └─7269 /usr/local/mysql/bin/mysqld
--basedir=/usr/local/mysql --datadir=/usr/local/mysql/data
--plugin-...
```

步骤 9 MySQL 基本设置

配置 MySQL 运行 root 远程连接：

```
> mysql -u root -p
Enter password: 输入步骤 8 自定义的 root 密码
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.7.27-log Source distribution

# 切换使用 mysql 数据库
mysql> use mysql;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed

# 设置远程登录主机与用户名
mysql> update user set host = '%' where user = 'root';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

# 查看设置结果
mysql> select host, user from user;
+-----------+---------------+
| host      | user          |
+-----------+---------------+
| %         | root          |
| localhost | mysql.session |
| localhost | mysql.sys     |
+-----------+---------------+
3 rows in set (0.00 sec)

# 刷新配置
mysql> flush privileges;
Query OK, 0 rows affected (0.01 sec)
```

步骤 10 安装 MySQL 的 java 连接工具

打开一个新的终端，在 node1 上，安装 mysql-connector-java：

```
> yum -y install mysql-connector-java
......
Installed:
  mysql-connector-java.noarch 1:5.1.25-3.el7


Dependency Installed:
  apache-commons-lang.noarch 0:2.6-15.el7
apache-commons-logging.noarch 0:1.1.2-7.el7
  avalon-framework.noarch 0:4.3-10.el7
avalon-logkit.noarch 0:2.1-14.el7
  cal10n.noarch 0:0.7.7-4.el7
geronimo-jms.noarch 0:1.1.1-19.el7
  geronimo-jta.noarch 0:1.1.1-17.el7                javamail.noarch
0:1.4.6-8.el7
  javassist.noarch 0:3.16.1-10.el7                  log4j.noarch
0:1.2.17-16.el7_4
  slf4j.noarch 0:1.7.4-4.el7_4
tomcat-servlet-3.0-api.noarch 0:7.0.76-11.el7_7
  xalan-j2.noarch 0:2.7.1-23.el7                    xerces-j2.noarch
0:2.11.0-17.el7_0
  xml-commons-apis.noarch 0:1.4.01-16.el7
xml-commons-resolver.noarch 0:1.2-15.el7


Complete!
```

4.3.2 部署 Hive

步骤 1 准备 Hive 组件包

在 node1 节点上执行如下命令，解压 Hive 软件包，并重命名文件夹：

```
> cp /home/extend_tools/apache-hive-2.3.3-bin.tar.gz /home/modules/
> cd /home/modules/
> tar -zvxf apache-hive-2.3.3-bin.tar.gz
> mv apache-hive-2.3.3-bin hive-2.3.3
> ls
```

步骤 2 拷贝 jar 连接文件到 Hive 指定目录中

```
> cp /usr/share/java/mysql-connector-java.jar
/home/modules/hive-2.3.3/lib/
```

步骤 3 配置 hive-site.xml

执行命令新建一个 hive-site.xml 文件：

```
> vim /home/modules/hive-2.3.3/conf/hive-site.xml
```

录入如下内容：

```
<configuration>
    <property>
```

```
        <name>javax.jdo.option.ConnectionURL</name>

<value>jdbc:mysql://node1:3306/hive_metadata?createDatabaseIfNotExis
t=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>root</value>
    </property>
    <property>
        <name>javax.jdo.option.ConnectionPassword</name>
        <value>输入 mysql 的 root 用户密码</value>
    </property>
    <property>
        <name>hive.strict.checks.cartesian.product</name>
        <value>false</value>
    </property>
</configuration>
```

注意：

1、配置文件中的 javax.jdo.option.ConnectionPassword 的值是 4.3.1 步骤 8 设置的 MySQL 里设置的 root 用户密码。

2、由于排版原因，javax.jdo.option.ConnectionURL 的 value 标签值分成了两行，实际为与其他 value 标签一样。

步骤 4 配置 hive-env.sh

在 node1 节点上复制一份 hive-env.sh 的模板文件，并修改它：

```
> cp /home/modules/hive-2.3.3/conf/hive-env.sh.template
/home/modules/hive-2.3.3/conf/hive-env.sh
> vim /home/modules/hive-2.3.3/conf/hive-env.sh
```

在末尾增加 HADOOP_HOME 路径

```
HADOOP_HOME=/home/modules/hadoop-2.8.3
```

步骤 5 初始化 MySQL 数据库

执行初始化 MySQL 数据库操作

```
> cd /home/modules/hive-2.3.3/bin
> ./schematool -initSchema -dbType mysql
......
[org.apache.logging.slf4j.Log4jLoggerFactory]
Metastore connection URL:
jdbc:mysql://node1:3306/hive_metadata?createDatabaseIfNotExist=true
```

```
Metastore Connection Driver :    com.mysql.jdbc.Driver
Metastore connection User:       root
Starting metastore schema initialization to 2.3.0
Initialization script hive-schema-2.3.0.mysql.sql
Initialization script completed
schemaTool completed
```

显示“schemaTool completed”则表示执行成功

步骤 6 分发 Hive 组件

在 node1 执行如下命令，将 Hive 目录拷贝到其他各个节点的/home/modules/下

```
> for i in {2..4};do scp -r /home/modules/hive-2.3.3
root@node${i}:/home/modules/;done
```

拷贝完毕后，在 node2~node4 节点，均可出现如下目录

```
> ls /home/modules/ |grep hive
hive-2.3.3
```

步骤 7 配置环境变量

在 node1~node4 上配置环境变量：

```
> vim /etc/profile

# 在末尾增加下面内容
export HIVE_HOME=/home/modules/hive-2.3.3
export PATH=$HIVE_HOME/bin:$PATH
```

然后在 node1~node4 节点执行如下命令确保环境变量生效：

```
> source /etc/profile
```

4.3.3 启动 Hive 并验证基本功能

在 node1 中执行命令 hive，打开 Hive 客户端：

```
> hive
Logging initialized using configuration in
jar:file:/home/modules/hive-2.3.3/lib/hive-common-2.3.3.jar!/hive-lo
g4j2.properties Async: true
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. spark, tez)
or using Hive 1.X releases.
hive>
```

在 Hive 客户端中执行查看数据库命令：

```
hive> show databases;
```

```
OK
default
Time taken: 4.224 seconds, Fetched: 1 row(s)
```

输出 default 表示命令执行成功，Hive 组件运行正常。

4.4 实验任务

4.4.1 Hive 创建表

4.4.1.1 建表语句

```
CREATE [EXTERNAL] TABLE [IF NOT EXISTS] table_name
  [(col_name data_type [COMMENT col_comment], ...)]
  [COMMENT table_comment]
  [PARTITIONED BY (col_name data_type [COMMENT col_comment], ...)]
  [CLUSTERED BY (col_name, col_name, ...)
  [SORTED BY (col_name [ASC|DESC], ...)] INTO num_buckets BUCKETS]
  [ROW FORMAT row_format]
  [STORED AS file_format]
  [LOCATION hdfs_path]
```

4.4.1.2 创建内部表

创建内部表 cga_info1，包含 name，gender 和 time 三列。

```
hive> create table cga_info1(name string,gender string,time int) row
format delimited fields terminated by ',' stored as textfile;
No rows affected (0.293 seconds)
```

其中，row format delimited fields terminated by ','表示行分隔符设置为'，'，如果不设置则使用默认分隔符。Hive 的 HQL 语句最后是以'；'结束的。

查看表“cga_info1”：

```
hive> show tables like 'cga_info1';
OK
cga_info1
Time taken: 0.028 seconds, Fetched: 1 row(s)
```

4.4.1.3 创建外部表

创建外部表时要注明 external table：

```
hive> create external table cga_info2 (name string,gender string,time
int) row format delimited fields terminated by ',' stored as textfile;
OK
Time taken: 0.058 seconds
```

查看创建表“cga_info2”：

```
hive> show tables like 'cga_info2';
```

4.4.1.4 载入本地数据

步骤 1 预先在本地/tmp 下建立一个文件。

新打开一个终端，在/root 路径下，然后新建一个文件：

```
> vim /root/hive.txt


# 添加一下内容：
xiaozhao,female,20
xiaoqian,male,21
xiaosun, male,25
xiaoli,female,40
xiaozhou,male,33
```

内容说明：内容有三列字段，分别为 name,gender,age。每一列的字段分隔符为英文逗号“,”：

步骤 2 建立一个名为“cga_info3”的表。

返回 hive 客户端，执行：

```
hive> create table cga_info3(name string,gender string,time int) row
format delimited fields terminated by ',' stored as textfile;
```

步骤 3 将本地数据“hive.txt”载入表“cga_info3”中。

```
> load data local inpath '/root/hive.txt' into table cga_info3;
```

步骤 4 查询表“cga_info3”中的内容。

```
hive> select * from cga_info3;
OK
xiaozhao        female  20
xiaoqian        male    21
xiaosun  male   25
xiaoli  female  40
xiaozhou        male    33
Time taken: 1.112 seconds, Fetched: 5 row(s)
```

由此可见，已经成功将本地文件“hive.txt”中的内容加载到了 Hive 表“cga_info3”中。

4.4.1.5 载入 HDFS 数据

步骤 1 首先在 HDFS 上创建文件夹“/tmp/hivetest”。

```
> hdfs dfs -mkdir -p /tmp/hivetest
```

步骤 2 将 tmp 文件夹中的本地文件“hive.txt”上传到 HDFS 的文件夹“/tmp/hivetest”。

```
> hdfs dfs -put /root/hive.txt /tmp/hivetest
```

步骤 3 查看是否上传成功。

```
> hdfs dfs -ls /tmp/hivetest


Found 1 items
-rw-r--r--   3 root supergroup        87 2020-10-15 22:36
/tmp/hivetest/hive.txt
```

步骤 4 返回 Hive 客户端。

建立一个名为“cga_info4”的表：

```
hive> create table cga_info4(name string,gender string,time int) row
format delimited fields terminated by ',' stored as textfile;
```

步骤 5 将 HDFS 文件“hive.txt”载入表“cga_info4”中。

```
hive> load data inpath '/tmp/hivetest/hive.txt' into table cga_info4;
Loading data to table default.cga_info4
OK
Time taken: 0.595 seconds
```

加载成功后，HDFS 的数据文件会被移动到/user/hive/warehouse 目录下，而原先/tmp/hivetest 路径下的 hive.txt 文件就没有了。

可以发现，加载本地数据和 HDFS 数据使用的命令略有不同：

```
加载本地文件：load data local inpath 'local_inpath' into table hive_table;
加载 HDFS 文件：load data inpath 'HDFS_inpath' into table hive_table。
```

步骤 6 查询表“cga_info4”中的内容。

```
hive> select * from cga_info4;
```

由此可见，已经将 HDFS 文件“hive.txt”的内容加载到了 Hive 表“cga_info4”中。

4.4.1.6 创建表时载入数据

重新上传文本文件到 hdfs

```
> hdfs dfs -put hive.txt /tmp/hivetest
```

创建表“cga_info5”同时载入 HDFS 上 hive.txt 的数据。

```
hive> create external table cga_info5 (name string,gender string,time
int) row format delimited fields terminated by ',' stored as textfile
location '/tmp/hivetest';
```

查询表“cga_info5”中的内容。

```
hive> select * from cga_info5;
```

由此可见，我们成功的在创建表“cga_info5”的同时载入了 HDFS 上 hive.txt 的数据。

4.4.1.7 复制一个空表

步骤 1 建立一个新表“cga_info6”,将表“cga_info1”复制到表“cga_info6”中。

```
hive> create table cga_info6 like cga_info1;
```

步骤 2 查询表“cga_info6”中的内容。

```
hive> select *from cga_info6;
```

结果为空，因为只是将表结构进行了复制，我们再来查看一下表结构：

```
hive> desc cga_info6;
OK
name                    string
gender                  string
time                    int
Time taken: 0.031 seconds, Fetched: 3 row(s)
```

由结果可知，复制空表成功。

4.4.2 查询

4.4.2.1 模糊查询表

查询以“cga 开头的表”。

```
hive> show tables like '*cga*';
OK
cga_info1
cga_info2
cga_info3
cga_info4
cga_info5
cga_info6
Time taken: 0.019 seconds, Fetched: 8 row(s)
```

4.4.2.2 条件查询

示例 1：使用 limit 查询表“cga_info3”中前两行的数据：

```
hive> select * from cga_info3 limit 2;
OK
xiaozhao        female  20
xiaoqian        male    21
Time taken: 0.094 seconds, Fetched: 2 row(s)
```

示例 2：使用 where 查询表“cga_info3”中所有女性的信息：

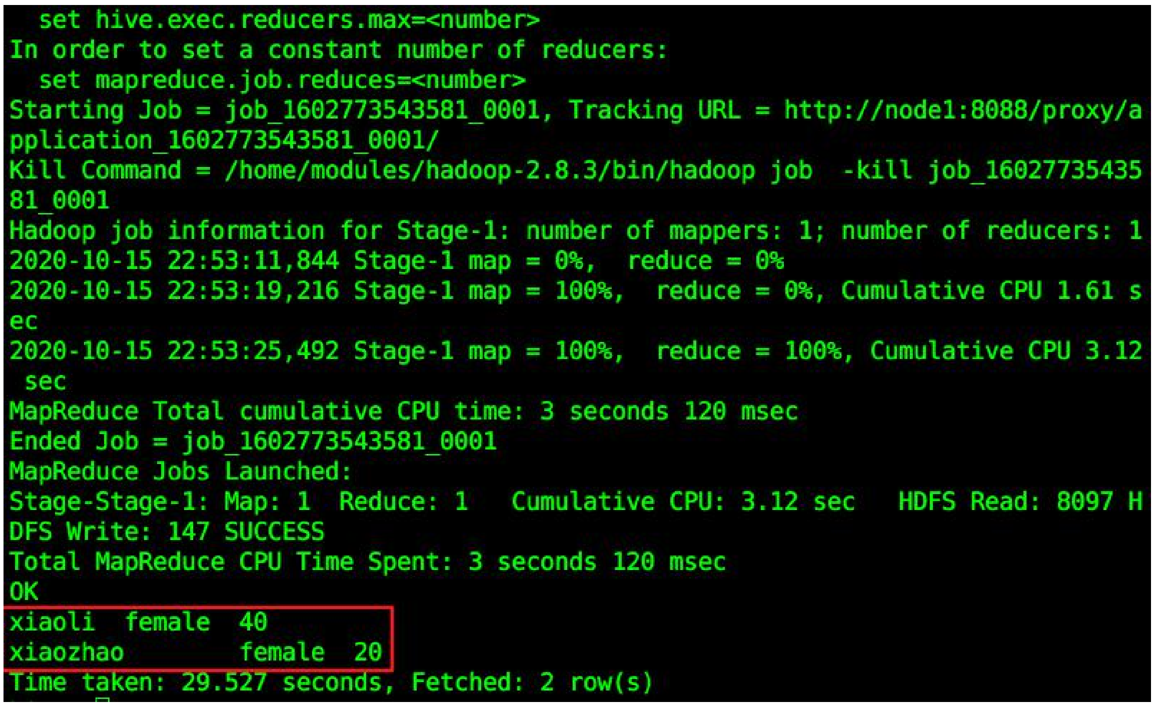
```
hive> select * from cga_info3 where gender='female';
OK
xiaozhao        female  20
xiaoli  female  40
Time taken: 0.397 seconds, Fetched: 2 row(s)
```

示例 3：使用 order 按时间递减顺序查询“cga_info3”中所有女性的信息：

```
hive> select * from cga_info3 where gender='female' order by time desc;
OK
xiaoli  female  40
xiaozhao        female  20
Time taken: 22.606 seconds, Fetched: 2 row(s)
```

由结果可以看出本来 xiaozhao 的信息是先输入的，查询结果按照 time 的递减排序，所以 xiaozhao 的信息排在了第二个。

注意：略微复杂的查询，可能会触发 Hive-on-MR 任务，输出类似如下的信息，这是正常现象：



```
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1602773543581_0001, Tracking URL = http://node1:8088/proxy/a
pplication_1602773543581_0001/
Kill Command = /home/modules/hadoop-2.8.3/bin/hadoop job  -kill job_16027735435
81_0001
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2020-10-15 22:53:11,844 Stage-1 map = 0%,  reduce = 0%
2020-10-15 22:53:19,216 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.61 s
ec
2020-10-15 22:53:25,492 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 3.12
 sec
MapReduce Total cumulative CPU time: 3 seconds 120 msec
Ended Job = job_1602773543581_0001
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 3.12 sec   HDFS Read: 8097 H
DFS Write: 147 SUCCESS
Total MapReduce CPU Time Spent: 3 seconds 120 msec
OK
xiaoli  female  40
xiaozhao        female  20
Time taken: 29.527 seconds, Fetched: 2 row(s)
```

4.4.2.3 多条件查询

示例 1：对表“cga_info3”进行查询，按姓名进行分组，找出 time 值大于 30 的人：

```
hive> select name,sum(time),all_time from cga_info3 group by name having all_time > 30;
OK
xiaoli  40
xiaozhou       33
Time taken: 17.133 seconds, Fetched: 2 row(s)
```

示例 2：对表“cga_info3”进行查询，按性别分组，找出 time 值最大的人：

```
hive> select gender,max(time) from cga_info3 group by gender;
OK
 male   25
female  40
male    33
Time taken: 16.013 seconds, Fetched: 3 row(s)
```

示例 3：统计表“cga_info3”中，女性和男性的总数各是多少：

```
hive> select gender,count(1) num from cga_info3 group by gender;
OK
 male   1
female  2
male    2
Time taken: 17.01 seconds, Fetched: 3 row(s)
```

4.4.2.4 复杂条件查询

将表“cga_info7”中女性的信息插入表”cga_info3”。

```
# 首先建立内部表“cga_info7”。
hive> create table cga_info7(name string,gender string,time int) row format delimited fields terminated by ',' stored as textfile;
```

步骤 1 在 node1 的/root 中创建本地文件“hive2.txt”并输入内容。

```
> vim hive2.txt


# 添加下面内容：
xiaozhao,female,20
xiaochen,female,28
```

步骤 2 将本地数据载入表“cga_info7”中。

```
hive> load data local inpath '/root/hive2.txt' into table cga_info7;
```

步骤 3 将表“cga_info7”中女性的信息加载到表“cga_info3”中。

```
hive> insert into cga_info3 select * from cga_info7 where gender='female';
```

步骤 4 查询表“cga_info3”中的内容。

```
hive> select * from cga_info3;
OK
xiaozhao        female  20
xiaochen        female  28
xiaozhao        female  20
xiaoqian        male    21
xiaosun  male   25
xiaoli  female  40
xiaozhou        male    33
Time taken: 0.069 seconds, Fetched: 7 row(s)
```

结果表明表“cga_info3”中增加了两条表“cga_info7 ”中女性信息。

示例 5：按姓名和性别分组，查询表“cga_info3”中每个人 time 值的总和：

```
hive> select name,gender,sum(time) time from cga_info3 group by
name,gender;
OK
xiaochen        female  28
xiaoli  female  40
xiaoqian        male    21
xiaosun  male   25
xiaozhao        female  40
xiaozhou        male    33
Time taken: 16.168 seconds, Fetched: 6 row(s)
```

从结果中可以看出，原本在表“cga_info3”中有两条 xiaozhao 的信息，现在已经被合并。

4.5 实验小结

本实验讲述了鲲鹏 BigData Pro 解决方案下 Hive 组件的安装部署，和 Hive 数据仓库的增删改查操作，使学员对 Hive 运维有更直观的认识。通过本章节实验，学员将增强对 Hive 的理解和使用。需要注意的是 load data 时，必须在创建表时指定 stored as textfile，否则数据将无法导入。