**Impala编译部署**

# 准备工作

## 1.1 操作系统

磁盘预留20G空间

### 1.1.1 挂载系统盘并设置为本地yum源

挂载系统盘（设备为/dev/cdrom）

mount -t auto /dev/cdrom /mnt

在/etc/yum.repo.d/下创建文件 cdrom.repo，内容为

[cdrom]

name=cdrom

baseurl=file:///mnt

gpgcheck=1

gpgkey=file:///mnt/RPM-GPG-KEY-CentOS-7

enabled=1

新建个bak目录，把其他的repo文件移动到bak目录下，优先使用cdrom安装。

### 1.1.2 安装JDK

最好是安装oracle JDK

如果安装openjdk-devel的包， 则需安装java-1.8.0-openjdk-devel包

yum install -y java-1.8.0-openjdk-devel

修改环境变量 ~/.bash\_profile

export JAVA\_HOME=/usr/lib/jvm/java-1.8.0-openjdk/

export PATH=$JAVA\_HOME/bin:$PATH

export CLASSPATH=.:$JAVA\_HOME/lib/dt.jar:$JAVA\_HOME/lib/tools.jar

### 1.1.3 安装lsb-release

yum install -y redhat-lsb.x86\_64

### 1.1.4 安装mvn

https://maven.apache.org/download.cgi

拷贝到服务器上

tar -xvf apache-maven-3.5.0-bin.tar.gz

mv apache-maven-3.5.0 /usr/local

设置环境变量

vim ~/.bash\_profile

export M2\_HOME=/usr/local/apache-maven-3.5.0

export M2=$M2\_HOME/bin

export PATH=$M2:$PATH

### 1.1.5 安装编译依赖

执行以下4条命令安装依赖：

yum groupinstall "Development Tools"

yum -y install git ant libevent-devel automake libtool flex bison gcc-c++ openssl-devel make cmake

yum -y install doxygen.x86\_64 glib-devel python-devel bzip2-devel svn libevent-devel krb5-workstation

yum -y install openldap-devel db4-devel python-setuptools python-pip cyrus-sasl\* postgresql postgresql-server ant-nodeps lzo-devel lzop

### 1.1.6 安装python依赖

下载get-pip.py

<https://bootstrap.pypa.io/get-pip.py>

执行以下2条命令

python get-pip.py

pip install allpairs pytest pytest-xdist paramiko texttable prettytable sqlparse psutil==0.7.1 pywebhdfs gitpython jenkinsapi boto3

### 1.1.7 设置环境变量

export IMPALA\_HOME=/root/impala

这个路径是impala源码根目录所对应的路径

## 1.2 源码下载

打开impala github ,选择分支 branch-2.9.0,下载为zip文件。然后将文件拷贝到编译服务器，/root目录下。

https://github.com/apache/incubator-impala

unzip incubator-impala-branch-2.9.0.zip

mv incubator-impala-branch-2.9.0 impala

# 编译impala

这一步在下载文件上面耗时较长，在执行编译之前，source ~/.bash\_profile使环境变量生效

cd /root/impala

./buildall.sh -notests

编译过程中需要下载各种依赖，耗时较长且如果网络不稳定会中断。在第一次编译时，如有网络中断则可以手动下载然后将文件拷贝到对应的目录，再继续编译过程。下载的URL可以在编译日志里面看到。注意保存好手动下载的文件。

有Python dependencies 和 toolchain denpendencies。

Python dependencies通常较小，一般没有问题。

Toolchain dependencies的目的目录为 ${IMPALA\_HOME}/toolchain/

# 拷贝编译后的文件

这一步的目的是将程序运行必须的文件拷贝出来，便于发布和部署使用，我只想测试kudu，这里只拷贝了kudu的库。

首先介绍下需要复制的内容

etc/default　　:准备放impala配置文件

etc/hadoop　　：hadoop配置文件

be　　　　　　：impala运行文件

lib64　　　 　：impala运行依赖库

hadoop　　　　：hadoop运行文件

hive ：hive运行文件

hbase ： hbase运行文件

dependency　　：jar包

shell　　　　　：impala-shell

www　　　　　：impala程序中web页面模板

复制以上内容可用的shell脚本，注意红色字体部分需要根据环境而定

*#!/bin/bash*

IMPALA\_HOME=/root/impala

IMPALA\_BUILD=/home/gwl/impala-build-2.10

*# if exist the dest directory then clear it.*

**if** [ -d "${IMPALA\_BUILD}" ]; **then**

rm -rf ${IMPALA\_BUILD}/\*

**else**

mkdir -p ${IMPALA\_BUILD}

**fi**

mkdir -p ${IMPALA\_BUILD}/etc/default/

mkdir -p ${IMPALA\_BUILD}/etc/hadoop/

mkdir -p ${IMPALA\_BUILD}/be

mkdir -p ${IMPALA\_BUILD}/lib64/

mkdir ${IMPALA\_BUILD}/hadoop/

mkdir ${IMPALA\_BUILD}/hive/

mkdir ${IMPALA\_BUILD}/hbase/

mkdir ${IMPALA\_BUILD}/dependency/

mkdir ${IMPALA\_BUILD}/shell

cp -rf ${IMPALA\_HOME}/be/build/debug/\* ${IMPALA\_BUILD}/be/

cp -rf ${IMPALA\_HOME}/toolchain/gcc-4.9.2/lib64/\* ${IMPALA\_BUILD}/lib64/

cp -rf ${IMPALA\_HOME}/fe/target/impala-frontend-0.1-SNAPSHOT.jar ${IMPALA\_BUILD}/dependency/

cp -rf ${IMPALA\_HOME}/fe/target/dependency/\* ${IMPALA\_BUILD}/dependency/

cp -rf ${IMPALA\_HOME}/toolchain/cdh\_components/hadoop-\*/\* ${IMPALA\_BUILD}/hadoop/

cp -rf ${IMPALA\_HOME}/toolchain/cdh\_components/hive-\*/\* ${IMPALA\_BUILD}/hive/

cp -rf ${IMPALA\_HOME}/toolchain/cdh\_components/hbase-\*/\* ${IMPALA\_BUILD}/hbase/

cp -rf ${IMPALA\_BUILD}/hadoop/etc/hadoop ${IMPALA\_BUILD}/etc/

cp -rf ${IMPALA\_HOME}/toolchain/kudu-\*/debug/lib64/\* ${IMPALA\_BUILD}/lib64/

cp -rf ${IMPALA\_HOME}/shell/build/impala-shell\*/\* ${IMPALA\_BUILD}/shell

cp -r ${IMPALA\_HOME}/www ${IMPALA\_BUILD}/

*# 修改catalog和statestore连接的指向，将其改为相对路径*

ln -snf ../service/impalad ${IMPALA\_BUILD}/be/catalog/catalogd

ln -snf ../service/impalad ${IMPALA\_BUILD}/be/statestore/statestored

echo "Finished"

# 单机部署

## 文件拷贝

将编译生成并复制的可执行文件，一起拷贝到本机某目录下，比如/root/impala2

## 操作系统

### 安装JDK

安装JDK。方法同编译部分。

### 环境变量

vim ~/.bashrc 加入如下

export JAVA\_HOME=/usr/lib/jdk1.8.0\_144

export PATH=$PATH:$JAVA\_HOME/bin

export CLASSPATH=.:$JAVA\_HOME/lib/dt.jar:$JAVA\_HOME/lib/tools.jar

vim ~/.bash\_profile

export IMPALA\_HOME=/usr/lib/impala

执行如下语句使更改生效

source /etc/default/impala

## 配置

### 配置Impala

将impala运行必须的配置文件放到/etc/default/impala中，大概内容如下

IMPALA\_CATALOG\_SERVICE\_HOST=127.0.0.1

IMPALA\_STATE\_STORE\_HOST=127.0.0.1

IMPALA\_STATE\_STORE\_PORT=24000

IMPALA\_BACKEND\_PORT=22000

IMPALA\_LOG\_DIR=/var/log/impala

export IMPALA\_STATE\_STORE\_ARGS=${IMPALA\_STATE\_STORE\_ARGS:- \

-log\_dir=${IMPALA\_LOG\_DIR} \

-state\_store\_port=${IMPALA\_STATE\_STORE\_PORT}}

export IMPALA\_SERVER\_ARGS=" \

-log\_dir=${IMPALA\_LOG\_DIR} \

-catalog\_service\_host=${IMPALA\_CATALOG\_SERVICE\_HOST} \

-state\_store\_port=${IMPALA\_STATE\_STORE\_PORT} \

-use\_statestore \

-state\_store\_host=${IMPALA\_STATE\_STORE\_HOST} \

-be\_port=${IMPALA\_BACKEND\_PORT}"

export ENABLE\_CORE\_DUMPS=${ENABLE\_COREDUMPS:-false}

export IMPALA\_CATALOG\_ARGS=" \

-catalog\_service\_host=${IMPALA\_CATALOG\_SERVICE\_HOST} \

-catalog\_service\_port=26000"

export IMPALA\_HOME=/usr/lib/impala

export HADOOP\_HOME="${IMPALA\_HOME}/../hadoop/"

export PATH=$PATH:${HADOOP\_HOME}/bin:${HADOOP\_HOME}/sbin

export LD\_LIBRARY\_PATH=$LD\_LIBRARY\_PATH:$HADOOP\_HOME/lib/native

export LD\_LIBRARY\_PATH=$IMPALA\_HOME/lib64/:$LD\_LIBRARY\_PATH

for f in $IMPALA\_HOME/dependency/\*.jar; do

export CLASSPATH=$CLASSPATH:$f

done

export MYSQL\_CONNECTOR\_JAR=/usr/lib/hive/lib/mysql-connector-java-5.1.43-bin.jar

export CLASSPATH=$CLASSPATH:${HADOOP\_HOME}/etc/hadoop/

export CATALOGCMD="${IMPALA\_HOME}/be/catalog/catalogd ${IMPALA\_CATALOG\_ARGS}"

export STATESTORECMD="${IMPALA\_HOME}/be/statestore/statestored ${IMPALA\_STATE\_STORE\_ARGS}"

export IMPALADCMD="${IMPALA\_HOME}/be/service/impalad ${IMPALA\_SERVER\_ARGS}"

### 配置Hadoop

mv ${IMPALA\_HOME}/etc/hadoop /etc

#### 配置hadoop-env.sh

在此文件的末尾加入 export JAVA\_HOME=/usr/java/jdk1.8.144

注意：jdk所在路径需视自己的环境而定

#### 配置yarn-env.sh

同hadoop-env.sh，在文件的末尾加入 export JAVA\_HOME=/usr/java/jdk1.8.144

注意：jdk所在路径需视自己的环境而定

#### 修改core-site.xml

<configuration>

<property>

<name>fs.default.name</name>

<value>hdfs://localhost:9000</value>

</property>

<property>

<name>hadoop.tmp.dir</name>

<value>/usr/lib/hadoop/tmp</value>

</property>

</configuration>

#### 修改hdfs-site.xml

<configuration>

<property>

<name>dfs.replicatioin</name>

<value>1</value>

</property>

<property>

<name>dfs.name.dir</name>

<value>/usr/lib/hadoop/hdfs/name</value>

</property>

<property>

<name>dfs.data.dir</name>

<value>/usr/lib/hadoop/hdfs/data</value>

</property>

<property>

<name>dfs.client.read.shortcircuit</name>

<value>true</value>

</property>

<property>

<name>dfs.domain.socket.path</name>

<value>/var/run/hdfs-sockets/dn</value>

</property>

<property>

<name>dfs.client.file-block-storage-locations.timeout.millis</name>

<value>10000</value>

</property>

<property>

<name>dfs.datanode.hdfs-blocks-metadata.enabled</name>

<value>true</value>

</property>

</configuration>

#### 修改mapred-site.xml,

在<configuration>标签间加入如下内容

<property>

<name>mapreduce.framework.name</name>

<value>yarn</value>

</property>

<property>

<name>mapreduce.jobhistory.address</name>

<value>172.16.22.53:10020</value>

</property>

<property>

<name>mapreduce.jobhistory.webapp.address</name>

<value>172.16.22.53:19888</value>

</property>

<property>

<name>yarn.app.mapreduce.am.staging-dir</name>

<value>/history</value>

</property>

<property>

<name>mapreduce.jobhistory.done-dir</name>

<value>${yarn.app.mapreduce.am.staging-dir}/history/done</value>

</property>

<property>

<name>mapreduce.jobhistory.intermediate-done-dir</name>

<value>${yarn.app.mapreduce.am.staging-dir}/history/done\_intermediate</value>

</property>

<property>

<name>mapreduce.map.log.level</name>

<value>DEBUG</value>

</property>

<property>

<name>mapreduce.reduce.log.level</name>

<value>DEBUG</value>

</property>

#### 修改yarn-site.xml

在<configuration>标签间加入如下内容

<property>

<name>mapreduce.framework.name</name>

<value>yarn</value>

</property>

<property>

<name>yarn.nodemanager.aux-services</name>

<value>mapreduce\_shuffle</value>

</property>

<property>

<name>yarn.resourcemanager.hostname</name>

<value>172.16.22.53</value>

</property>

<property>

<name>yarn.resourcemanager.address</name>

<value>172.16.22.53:8032</value>

</property>

<property>

<name>yarn.resourcemanager.scheduler.address</name>

<value>172.16.22.53:8030</value>

</property>

<property>

<name>yarn.resourcemanager.resource-tracker.address</name>

<value>172.16.22.53:8031</value>

</property>

<property>

<name>yarn.resourcemanager.admin.address</name>

<value>172.16.22.53:8033</value>

</property>

<property>

<name>yarn.resourcemanager.webapp.address</name>

<value>172.16.22.53:8088</value>

</property>

<property>

<name>yarn.log-aggregation-enable</name>

<value>true</value>

</property>

<property>

<name>yarn.log.server.url</name>

<value>http://172.16.22.53:19888/jobhistory/logs</value>

</property>

#### hadoop初始化

mkdir -p /usr/lib/hadoop/hdfs/name

mkdir -p /usr/lib/hadoop/hdfs/data

mkdir /var/run/hdfs-sockets/

source /etc/hadoop/hadoop-env.sh

hadoop namenode -format

#### impala初始化

mkdir /var/log/impala

### 配置Hive



#### 修改配置文件

##### 修改hive-site.xml配置文件

执行命令：

cd $HIVE\_HOME/conf

cp hive-default.xml.template hive-site.xml

vi hive-site.xml # 红色字体部分需要根据自己的环境进行配置

删除<configuration>标签下的所有内容，并加入以下内容

<!--增加这两行，在使用Mysql时 -->

<property>

<name>system:java.io.tmpdir</name>

<value>/usr/lib/apache-hive-1.2.2-bin/iotmp</value>

<description/>

</property>

<property>

<name>system:user.name</name>

<value>username</value>

<description />

</property>

<!--增加这两行，在使用Mysql时 -->

<property>

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

<value>jdbc:mysql://172.22.16.53:3306/hive?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>hive</value>

<description>username to use against metastore database</description>

</property>

<property>

<name>javax.jdo.option.ConnectionPassword</name>

<value>hive</value>

<description>password to use against metastore database</description>

</property>

<property>

<name>hive.server2.thrift.port</name>

<value>10000</value>

</property>

<property>

<name>hive.server2.thrift.bind.host</name>

<value>172.22.16.53</value>

</property>

##### 修改hive-log4j.properties配置文件

执行

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

vi hive-log4j.properties

将hive.log.dir的值修改为/var/log/hive

##### 修改hive-env.sh配置文件

执行

cp hive-env.sh.template hive-env.sh

vi hive-env.sh

配置Hive配置文件的路径和Hadoop的安装路径

export HIVE\_CONF\_DIR=/usr/lib/apache-hive-1.2.2-bin/conf

HADOOP\_HOME=/usr/lib/hadoop-2.6.0

#### 将mysql的驱动放置到hive的lib下

下载<https://dev.mysql.com/get/Downloads/Connector-J/mysql-connector-java-5.1.43.tar.gz>

解压到hive的lib目录下，执行如下命令

tar -xzvf mysql-connector-java-5.1.43.tar.gz

mv mysql-connector-java-5.1.43/mysql-connector-java-5.1.43-bin.jar $HIVE\_HOME/lib

#### 在mysql中创建hive用户及数据库

mysql -uroot –proot 登录mysql

mysql> create user 'hive' identified by 'hive';

mysql> grant all privileges on \*.\* to 'hive' with grant option;

mysql> grant all privileges on \*.\* to 'hive' @'localhost' identified by 'hive' with grant option;

mysql> flush privileges;

修改mysql字符集

mysql>alter database hive character set latin1;

#### 替换jline的jar包

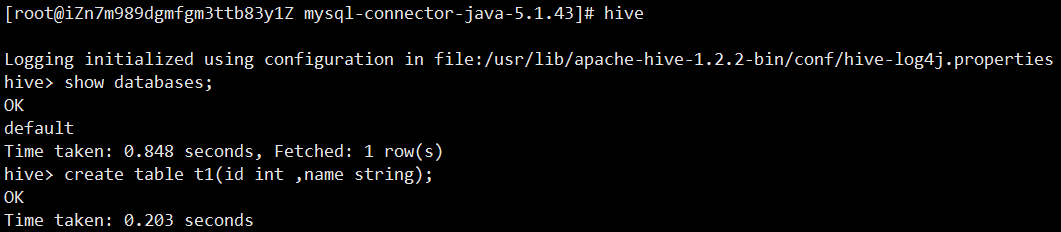
将/usr/lib/hadoop-2.6.0/share/hadoop/yarn/lib 下的jline-0.9.94.jar 移除，并将/usr/lib/apache-hive-1.2.1/lib 下的jline-2.12.jar 复制到该文件夹下，执行如下命令

rm -f $HADOOP\_HOME/share/hadoop/yarn/lib/jline-0.9.94.jar

mv $HIVE\_HOME/lib/jline-2.12.jar $HADOOP\_HOME/share/hadoop/yarn/lib/

#### 启动hive

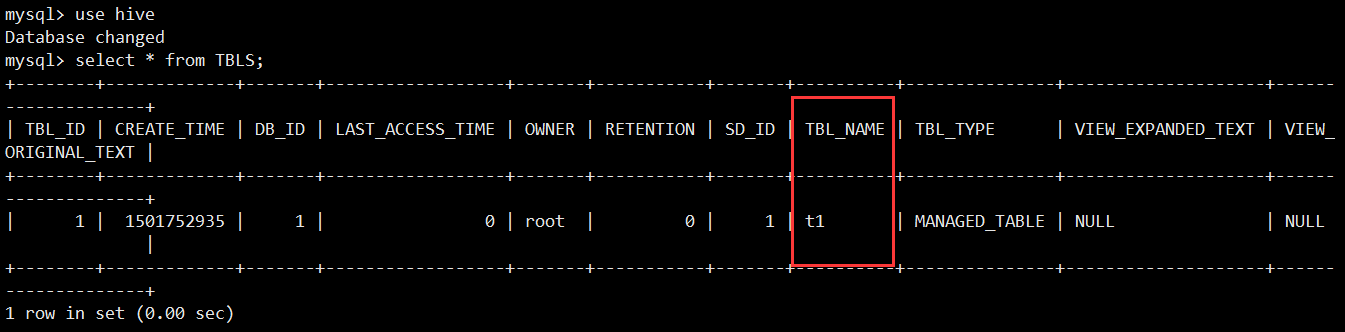
1. 命令行输入hive 即可进入hive的命令行
2. 创建表，以验证hive配置正确



1. 登录mysql 查看创建的表

执行 use hive;

select \* from TBLS; 查看创建的table



## 运行

### 启动hdfs

执行如下命令

source /etc/hadoop/hadoop-env.sh

start-dfs.sh

调用jps应该能看到有NameNode，DataNode进程。

### 启动impala服务

1. Catalogd

nohup ${CATALOGCMD} >> catalog.log 2>&1 &

1. Statestored

nohup ${STATESTORECMD} >> statestore.log 2>&1 &

1. Impalad

nohup ${IMPALADCMD} >> impalad.log 2>&1 &

这里Impalad指定了使用的kudu的master地址。如果不需要可以不写参数 kudu\_master\_hosts。

注意：

在这里指定了kudu master的地址。需要在本机/etc/hosts中配置 kudu master。

比如 192.168.179.146 kudu

## shell功能测试

1. 打开shell

${IMPALA\_HOME}/shell/impala-shell

1. 查看版本

select version();

1. 查看数据库

show databases;

1. 查看表

show tables;

1. 创建表

create table test1

(

uid bigint primary key,

uname string

)

partition by hash(uid) partitions 2 stored as kudu;

1. 创建外部表

假定在kudu上已有了一个表名为test1，在impala中创建一个表test12和这个test1做映射。

CREATE EXTERNAL TABLE IF NOT EXISTS test12 COMMENT 'test1'

STORED AS KUDU

TBLPROPERTIES ('kudu.table\_name'='impala::default.test1');

退出shell

Quit;

# 集群部署

## 准备集群主机

规划各主机的IP地址，主机名，每主机上要部署的服务。例如

|  |  |  |  |
| --- | --- | --- | --- |
| IP地址 | 主机名 | Hadoop服务 | Impala服务 |
| 192.168.179.150 | Impala1 | NameNode,DataNode | Impalad |
| 192.168.179.151 | Impala2 | DataNode | Impalad,  Catalogd,Statestored |
| 192.168.179.152 | Impala3 | DataNode | Impalad |

备注：Hadoop 实现单NameNode，三个DataNode模式。如果NameNode采用主备模式，则下面的配置需要更改，步骤也不尽相同。Hadoop NameNode配置不是本文重点。单NameNode已足够集群功能演练使用。

## SSH免密登录

## 各个节点部署

### 文件拷贝

将编译生成并复制的可执行文件，一起拷贝到本机某目录下，比如/root/impala2

### 安装JDK

### 配置环境变量

~/.bashrc

export JAVA\_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.65-3.b17.el7.x86\_64

export PATH=$PATH:$JAVA\_HOME/bin

export CLASSPATH=.:$JAVA\_HOME/lib/dt.jar:$JAVA\_HOME/lib/tools.jar

~/.bash\_profile

export IMPALA\_HOME=/root/impala2

source /etc/default/impala

### 关闭防火墙

systemctl disable firewalld

systemctl stop firewalld

### 关闭selinux

### 修改hosts文件

192.168.179.150 impala1

192.168.179.151 impala2

192.168.179.152 impala3

### 时间同步NTP

### HDFS配置

操作需要在每台主机上执行，配置文件core-site.xml,hdfs-site.xml 各主机要保持一致。

在每个datanode执行

mkdir /var/run/hdfs-sockets/

mkdir -p /usr/local/hadoop/hdfs/data

mkdir /var/run/hdfs-sockets/

在namenode执行

mkdir -p /usr/local/hadoop/hdfs/name

#### 修改core-site.xml

文件路径 /etc/hadoop/core-site.xml

<configuration>

<property>

<name>fs.default.name</name>

<value>hdfs://impala1</value>

</property>

<property>

<name>hadoop.tmp.dir</name>

<value>/usr/local/hadoop/tmp</value>

</property>

#### 修改hdfs-site.xml

文件路径 /etc/hadoop/hdfs-site.xml

<configuration>

<property>

<name>dfs.nameservices</name>

<value>impala1</value>

</property>

<property>

<name>dfs.ha.namenodes.impala1</name>

<value>impala1</value>

</property>

<property>

<name>dfs.replicatioin</name>

<value>2</value>

</property>

<property>

<name>dfs.name.dir</name>

<value>/usr/local/hadoop/hdfs/name</value>

</property>

<property>

<name>dfs.data.dir</name>

<value>/usr/local/hadoop/hdfs/data</value>

</property>

<property>

<name>dfs.client.read.shortcircuit</name>

<value>true</value>

</property>

<property>

<name>dfs.domain.socket.path</name>

<value>/var/run/hdfs-sockets/dn</value>

</property>

<property>

<name>dfs.client.file-block-storage-locations.timeout.millis</name>

<value>10000</value>

</property>

<property>

<name>dfs.datanode.hdfs-blocks-metadata.enabled</name>

<value>true</value>

</property>

</configuration>

#### 修改slaves

配置hdfs datanode 主机名，文件/etc/hadoop/slaves

内容为

Impala1

Impala2

Impala3

### HDFS初始化

在namenode上执行

source /etc/hadoop/hadoop-env.sh

hadoop namenode -format

## Impala配置

操作需要在每台主机上执行，配置文件各主机保持一致。

### Impala配置文件

路径/etc/default/impala

IMPALA\_STATE\_STORE\_HOST=impala2

IMPALA\_STATE\_STORE\_PORT=24000

IMPALA\_BACKEND\_PORT=22000

IMPALA\_LOG\_DIR=/var/log/impala

IMPALA\_CATALOG\_SERVICE\_HOST=impala2

其他和单机一样

### 初始化

mkdir /var/log/impala

### 启动服务

修改~/.bash\_profile增加 避免每次输入

source /etc/hadoop/hadoop-env.sh

注意在启动前source /etc/default/impala 使配置生效。

#### 启动HDFS

在namenode上执行

source /etc/hadoop/hadoop-env.sh

start-dfs.sh

启动成功后jps可以看到每主机上都有datanode，在impala1上有namenode进程。

## 启动impala服务

impala2

nohup ${CATALOGCMD} >> /var/log/impala/catalog.log 2>&1 &

nohup ${STATESTORECMD} >> /var/log/impala/statestore.log 2>&1 &

所有主机上

nohup ${IMPALADCMD} >>/var/log/impala/impalad.log 2>&1 &

## Shell功能测试

可以在任何一台运行了impalad的机器上打开impala-shell。

功能测试和单机部署时的shell功能测试一样

Shell功能测试

不同主机上shell连接的实际是同一个impala集群。所以在A主机上创建的表，可以在B主机上的shell中看到并操作。

## web服务监控

Impala各组件提供了web监控的方式，可以以此查看配置和运行情况。

http://192.168.179.151:25000/

http://192.168.179.151:25010/

http://192.168.179.151:25020/

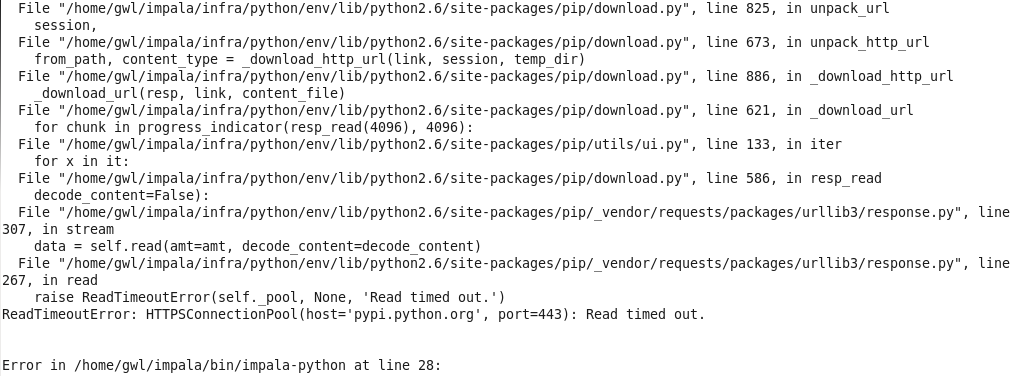
# 附录1：出现的问题

## gc 相关报错

执行 yum install gcc-c++ python-devel.x86\_64 cyrus-sasl-devel.x86\_64

## 连接失败

问题如图所示：



解决方法，检查端口是否被占用，然后重新启动编译脚本

## libjsig.so缺失

报错信息：

catalog/catalogd: error while loading shared libraries: libjsig.so: cannot open shared object file: No such file or directory

解决方法：

1. 在/etc/ld.so.conf.d/ 目录下新建文件impala-lib.conf 文件中加入如下内容

/usr/java/jdk1.8.0\_131/jre/lib/amd64/

/usr/java/jdk1.8.0\_131/jre/lib/amd64/server/

/usr/lib/impala/lib64/

1. 执行 ldconfig 既可解决