链滴

CentOS7 安装 Hbase-0.98.24

作者：k8s

原文链接：https://ld246.com/article/1589702665002

来源网站：链滴

0. 准备安装环境

继 CentOS7 安装 hadoop-1.2.1 准备安装环境

1. 安装 Hbase

1.1 安装到指定目录

下载 Hbase 安装包到 **/usr/local/src** 目录下。

https://archive.apache.org/dist/hbase/0.98.24/hbase-0.98.24-hadoop1-bin.tar.gz

```
[root@master /usr/local/src]# ls hbase-0.98.24-hadoop1-bin.tar.gz
hbase-0.98.24-hadoop1-bin.tar.gz
```

解压到/usr/local 下

```
[root@master /usr/local]# tar zxf src/hbase-0.98.24-hadoop1-bin.tar.gz
[root@master /usr/local]# ls
bin  etc  games  hadoop-1.2.1  hbase-0.98.24-hadoop1  include  jdk1.8.0_212  lib  lib64  libexec  sbin  share  src  zookeeper-3.4.5
```

1.2 配置服务

进入 **/usr/local/hbase-0.98.24-hadoop1/conf** 目录

配置 hbase-site.xml 文件

```
[root@master /usr/local/hbase-0.98.24-hadoop1/conf]# vim hbase-site.xml
...
<configuration>
    <!--用来指定使用hbase时产生文件的存放目录-->
    <property>
        <name>hbase.tmp.dir</name>
        <value>/var/hbase</value>
    </property>
    <!--指定hbase在hdfs里的根目录-->
    <property>
        <name>hbase.rootdir</name>
        <value>hdfs://master:9000/hbase</value>
    </property>
    <!--指定是否采用分布式模式-->
    <property>
        <name>hbase.cluster.distributed</name>
        <value>true</value>
    </property>
    <!--配置zookeeper集群节点地址-->
    <property>
        <name>hbase.zookeeper.quorum</name>
        <value>master,slave1,slave2</value>
    </property>
    <!--指定存储zookeeper快照信息的目录-->
    <property>
        <name>hbase.zookeeper.property.dataDir</name>
        <value>/usr/local/hbase-0.98.24-hadoop1/zookeeper</value>
    </property>
</configuration>
```

配置 hbase-env.sh

```
[root@master /usr/local/hbase-0.98.24-hadoop1/conf]# vim hbase-env.sh
...
# The java implementation to use.  Java 1.6 required.
export JAVA_HOME=/usr/local/jdk1.6.0_45
export CLASSPATH=.:$CLASSPATH:$JAVA_HOME/lib
...
# Tell HBase whether it should manage it's own instance of Zookeeper or not.
export HBASE_MANAGES_ZK=true
```

配置 regionservers 文件

```
[root@master /usr/local/hbase-0.98.24-hadoop1/conf]# cat regionservers
master
slave1
slave2
```

同步 /usr/local/hbase-0.98.24-hadoop1 到 slave1、slavd2

```
rsync -avzP /usr/local/hbase-0.98.24-hadoop1 root@slave1:/usr/local/
rsync -avzP /usr/local/hbase-0.98.24-hadoop1 root@slave2:/usr/local/
```

编辑 /etc/bashrc 文件配置环境变量

三个节点都要配置

```
[root@master ~]# tail -n18 /etc/bashrc

# java conf
export JAVA_HOME=/usr/local/jdk1.6.0_45

# hadoop conf
export HADOOP_HOME=/usr/local/hadoop-1.2.1

# zookeeper conf
export ZOOKEEPER_HOME=/usr/local/zookeeper-3.4.5

# Hbase conf
export HBASE_HOME=/usr/local/hbase-0.98.24-hadoop1
export HBASE_CLASSPATH=$HBASE_HOME/conf
export HBASE_LOG_DIR=$HBASE_HOME/logs

export CLASSPATH=.:$CLASSPATH:$JAVA_HOME/lib
export PATH=$JAVA_HOME/bin:/usr/local/hadoop-1.2.1/bin:$ZOOKEEPER_HOME/bin:$HBAS
```

```
_HOME/bin:$PATH
```

```
[root@master ~]# source /etc/bashrc
```

2. 启动 Hbase

因此处使用 hbase 自带 zookeeper，所以需要停掉之前安装的 zookeeper

在主节点启动 Hbase

```
[root@master ~]# start-hbase.sh
slave1: starting zookeeper, logging to /usr/local/hbase-0.98.24-hadoop1/logs/hbase-root-zo
keeper-slave1.out
master: starting zookeeper, logging to /usr/local/hbase-0.98.24-hadoop1/logs/hbase-root-zo
keeper-master.out
slave2: starting zookeeper, logging to /usr/local/hbase-0.98.24-hadoop1/logs/hbase-root-zo
keeper-slave2.out
slave1: SLF4J: Class path contains multiple SLF4J bindings.
slave1: SLF4J: Found binding in [jar:file:/usr/local/hbase-0.98.24-hadoop1/lib/slf4j-log4j12-1.6
4.jar!/org/slf4j/impl/StaticLoggerBinder.class]
slave1: SLF4J: Found binding in [jar:file:/usr/local/hadoop-1.2.1/lib/slf4j-log4j12-1.4.3.jar!/org/
lf4j/impl/StaticLoggerBinder.class]
slave1: SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
master: SLF4J: Class path contains multiple SLF4J bindings.
master: SLF4J: Found binding in [jar:file:/usr/local/hbase-0.98.24-hadoop1/lib/slf4j-log4j12-1.6
4.jar!/org/slf4j/impl/StaticLoggerBinder.class]
master: SLF4J: Found binding in [jar:file:/usr/local/hadoop-1.2.1/lib/slf4j-log4j12-1.4.3.jar!/org
slf4j/impl/StaticLoggerBinder.class]
master: SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
slave2: SLF4J: Class path contains multiple SLF4J bindings.
slave2: SLF4J: Found binding in [jar:file:/usr/local/hbase-0.98.24-hadoop1/lib/slf4j-log4j12-1.6
4.jar!/org/slf4j/impl/StaticLoggerBinder.class]
slave2: SLF4J: Found binding in [jar:file:/usr/local/hadoop-1.2.1/lib/slf4j-log4j12-1.4.3.jar!/org/
lf4j/impl/StaticLoggerBinder.class]
slave2: SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
starting master, logging to /usr/local/hbase-0.98.24-hadoop1/logs/hbase-root-master-master
out
slave2: starting regionserver, logging to /usr/local/hbase-0.98.24-hadoop1/logs/hbase-root-r
gionserver-slave2.out
slave1: starting regionserver, logging to /usr/local/hbase-0.98.24-hadoop1/logs/hbase-root-r
gionserver-slave1.out
master: starting regionserver, logging to /usr/local/hbase-0.98.24-hadoop1/logs/hbase-root-
egionserver-master.out
```

```
[root@master ~]# jps
9101 JobTracker
9010 SecondaryNameNode
9519 HQuorumPeer
8839 NameNode
10125 Jps
9640 HMaster
9797 HRegionServer
```

原文链接：CentOS7 安装 Hbase-0.98.24

```
[root@slave1 ~]# jps
9074 HRegionServer
9293 Jps
8818 TaskTracker
8720 DataNode
8950 HQuorumPeer
```

```
[root@slave2 ~]# jps
8759 DataNode
8857 TaskTracker
9348 Jps
8990 HQuorumPeer
9120 HRegionServer
```

3. hbase shell 检查服务状态

进入 hbase shell

```
[root@master ~]# hbase shell
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 0.98.24-hadoop1, r9c13a1c3d8cf999014f30104d1aa9d79e74ca3d6, Thu Dec 22 02:28:
5 UTC 2016

hbase(main):001:0>
```

检查状态

```
hbase(main):001:0> status
...
1 active master, 0 backup masters, 3 servers, 0 dead, 0.6667 average load
```

在此笔者遇到了连接不到 master 的 60000 端口的问题。

```
ERROR: org.apache.hadoop.hbase.ipc.ServerNotRunningYetException: Server master/10.2.96.3
60000 is not running yet
```

经排查，发现 60000 端口监听在了 ipv6 的 IP 上，禁用 ipv6

4. 安装 Thrift Server

下载 thrift 安装包到**/usr/local/src**目录下。

https://archive.apache.org/dist/thrift/0.8.0/thrift-0.8.0.tar.gz

```
[root@master /usr/local/src]# ls thrift-0.8.0.tar.gz
thrift-0.8.0.tar.gz
```

准备编译环境

```
[root@master ~]# yum -y install automake libtool flex bison pkgconfig gcc-c++ boost-devel l
```

```
bevent-devel zlib-devel python-devel ruby-devel openssl-devel
```

安装 Thrift

```
[root@master /usr/local/src]# tar zxf thrift-0.8.0.tar.gz
[root@master /usr/local/src]# cd thrift-0.8.0
[root@master /usr/local/src/thrift-0.8.0]# ./configure --with-cpp=no --with-ruby=no
...
thrift 0.8.0

Building code generators ..... :

Building C++ Library ......... : no
Building C (GLib) Library .... : no
Building Java Library ........ : no
Building C# Library .......... : no
Building Python Library ...... : yes
Building Ruby Library ........ : no
Building Haskell Library ..... : no
Building Perl Library ........ : no
Building PHP Library ......... : no
Building Erlang Library ...... : no
Building Go Library .......... : no

Using Python ................. : /usr/bin/python

If something is missing that you think should be present,
please skim the output of configure to find the missing
component.  Details are present in config.log.
[root@master /usr/local/src/thrift-0.8.0]# echo $?
0
[root@master /usr/local/src/thrift-0.8.0]# make
make  all-recursive
make[1]: 进入目录“/usr/local/src/thrift-0.8.0”
Making all in compiler/cpp
make[2]: 进入目录“/usr/local/src/thrift-0.8.0/compiler/cpp”
make  all-am
make[3]: 进入目录“/usr/local/src/thrift-0.8.0/compiler/cpp”
...
make[3]: 离开目录“/usr/local/src/thrift-0.8.0/test”
make[2]: 离开目录“/usr/local/src/thrift-0.8.0/test”
make[2]: 进入目录“/usr/local/src/thrift-0.8.0”
make[2]: 离开目录“/usr/local/src/thrift-0.8.0”
make[1]: 离开目录“/usr/local/src/thrift-0.8.0”
[root@master /usr/local/src/thrift-0.8.0]# echo $?
0
[root@master /usr/local/src/thrift-0.8.0]# make install
...
make[2]: 对“install-exec-am”无需做任何事。
make[2]: 对“install-data-am”无需做任何事。
make[2]: 离开目录“/usr/local/src/thrift-0.8.0”
make[1]: 离开目录“/usr/local/src/thrift-0.8.0”
[root@master /usr/local/src/thrift-0.8.0]# echo $?
0
```

启动 Thrift server

```
[root@master ~]# hbase-daemon.sh start thrift
starting thrift, logging to /usr/local/hbase-0.98.24-hadoop1/logs/hbase-root-thrift-master.ou

[root@master ~]# jps
9101 JobTracker
21649 ThriftServer
9010 SecondaryNameNode
9519 HQuorumPeer
8839 NameNode
9640 HMaster
21743 Jps
9797 HRegionServer
```

5. 安装 Thrift api for python

产生针对 Python 的 Hbase 的 API

下载 hbase 源码：

https://archive.apache.org/dist/hbase/0.98.24/hbase-0.98.24-src.tar.gz

```
[root@master /usr/local/src]# ls hbase-0.98.24-src.tar.gz
hbase-0.98.24-src.tar.gz
```

解压寻找 Hbase.thrift 文件

```
[root@master /usr/local/src]# tar zxf hbase-0.98.24-src.tar.gz
[root@master /usr/local/src]# cd hbase-0.98.24
[root@master /usr/local/src/hbase-0.98.24]# find . -name Hbase.thrift
./hbase-thrift/src/main/resources/org/apache/hadoop/hbase/thrift/Hbase.thrift
```

生成 python 的 hbase 模块

```
[root@master /usr/local/src/hbase-0.98.24]# cd ./hbase-thrift/src/main/resources/org/apache
hadoop/hbase/thrift
[root@master /usr/local/src/hbase-0.98.24/hbase-thrift/src/main/resources/org/apache/had
op/hbase/thrift]# ls
Hbase.thrift
[root@master /usr/local/src/hbase-0.98.24/hbase-thrift/src/main/resources/org/apache/had
op/hbase/thrift]# thrift -gen py Hbase.thrift
[root@master /usr/local/src/hbase-0.98.24/hbase-thrift/src/main/resources/org/apache/had
op/hbase/thrift]# ls gen-py
hbase  __init__.py
```

安装 hbase 模块

```
[root@master /usr/local/src/hbase-0.98.24/hbase-thrift/src/main/resources/org/apache/had
op/hbase/thrift]# cp -raf gen-py/hbase /usr/lib64/python2.7/site-packages/
```

Hbase 的 Python 操作

实例化连接

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from thrift import Thrift
from thrift.transport import TSocket
from thrift.transport import TTransport
from thrift.protocol import TBinaryProtocol

from hbase import Hbase
from hbase.ttypes import *

transport = TSocket.TSocket('master', 9090)
transport = TTransport.TBufferedTransport(transport)

protocol = TBinaryProtocol.TBinaryProtocol(transport)

client = Hbase.Client(protocol)

transport.open()

#=========
```

实例 1：创建表

```
#=========

base_info_contents = ColumnDescriptor(name='meta_data:', maxVersions=1)
other_info_contents = ColumnDescriptor(name='flags:', maxVersions=1)

client.createTable('new_music_table', [base_info_contents, other_info_contents])

print client.getTableNames()

[root@master ~/hbase_test]# python create_table.py
['music_table', 'new_music_table']
```

实例 2：插入数据

```
#=========
tableName = 'new_music_table'
rowKey = '1100'

mutations = [Mutation(column="meta_data:name", value="wangqingshui"), \
       Mutation(column="meta_data:tag", value="pop"), \
       Mutation(column="flags:is_valid", value="TRUE")]

client.mutateRow(tableName, rowKey, mutations, None)
```

实例 3：读取多条记录

```
#========
tableName = 'new_music_table'

scan = TScan()
id = client.scannerOpenWithScan(tableName, scan, None)
result = client.scannerGetList(id, 10)

for r in result:
    print '======'
    print 'the row is ' , r.row

    for k, v in r.columns.items():
        print "\t".join([k, v.value])
```

实例 4：读取指定 row key 记录

```
#========
tableName = 'new_music_table'
rowKey = '1100'

result = client.getRow(tableName, rowKey, None)

for r in result:
    print 'the row is ' , r.row
    print 'the name is ' , r.columns.get('meta_data:name').value
    print 'the flag is ' , r.columns.get('flags:is_valid').value
```

原文链接：CentOS7 安装 Hbase-0.98.24