系统:centos-7-x86\_64

1. 安装前设置
2. 若是GNOME桌面，且为中文目录

export LANG=en\_US

xdg-user-dirs-update

export LANG=zh\_CN.UTF-8

1. 网络设置

hostname:master , slave1 , slave2

vim /etc/hosts

192.168.1.131 master

192.168.1.132 slave1

192.168.1.133 slave2

# hostnamectl set-hostname master    (CentOS7 下原有的修改host方法无效了)

修改 /etc/hostname

静态ip配置：

vim /etc/sysconfig/network-scripts/ifcfg-eth0

修改文件如下

BOOTPROTO=”STATIC”

ONBOOT=”yes” #开机启动本设置

IPADDR=192.168.1.131

GATEWAY=192.168.1.1

NETMASK=255.255.255.0

DNS1=8.8.8.8

关闭防火墙:

# systemctl status firewalld.service  --查看防火墙状态

# systemctl stop firewalld.service    --关闭防火墙

# systemctl disable firewalld.service --永久关闭防火墙

1. 检测SSH安装情况，如果没有则安装SSH
2. 创建hadoop用户

# useradd hadoop --创建用户名为hadoop的用户# passwd hadoop  --为用户hadoop设置密码

1. 配置SSH无密钥登陆

**-----------下面是在master上面的操作**

# su hadoop --切换到hadoop用户$ cd ~      --打开用户文件夹

# ssh-keygen -t rsa -P ''

$ had --生成密码对，/home/hadoop/.ssh/id\_rsa和/home/hadoop/.ssh/id\_rsa.pub

$ cat ~/.ssh/id\_rsa.pub >> ~/.ssh/authorized\_keys        --把id\_rsa.pub追加到授权的key里面去

$ chmod 600 ~/.ssh/authorized\_keys        --修改权限

$ su  --切换到root用户# vim /etc/ssh/sshd\_config   --修改ssh配置文件

 RSAAuthentication yes #启用RSA认证

 PubkeyAuthentication yes #启用公钥私钥配对认证方式

 AuthorizedKeysFile .ssh/authorized\_keys #公钥文件路径

# su hadoop --切换到hadoop用户

$ scp ~/.ssh/authorized\_keys hadoop@192.168.1.8:~/                --把公钥复制所有的Slave机器上

**----------下面是在slave1上面的操作**

 # su hadoop --切换到hadoop用户

 $ mkdir ~/.ssh

 $ chmod 700 ~/.ssh

$ ssh-keygen -t rsa -P ''

 $ cat ~/id\_rsa.pub >> ~/.ssh/authorized\_keys                --追加到授权文件"authorized\_keys"

 $ chmod 600 ~/.ssh/authorized\_keys                          --修改权限

 $ su --切换回root用户 # vim /etc/ssh/sshd\_config   --修改ssh配置文件

  RSAAuthentication yes #启用RSA认证

  PubkeyAuthentication yes #启用公钥私钥配对认证方式

  AuthorizedKeysFile .ssh/authorized\_keys #公钥文件路径

1. 安装必须的软件
2. 安装JDK（jdk1.8编译hadoop时会出问题）

Chmod –R 777 /opt/

tar -zxvf jdk-8u45-linux-x64.tar.gz -C /opt/

mv jdk1.7.0\_79/ java

vim /etc/profile 加入以下内容

export JAVA\_HOME=/opt/java

export JRE\_HOME=${JAVA\_HOME}/jre

export CLASSPATH=.:${JAVA\_HOME}/lib:${JRE\_HOME}/lib

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

再用以下命令配置默认JDK版本

update-alternatives --install /usr/bin/java java /usr/opt/java/bin/java 300

update-alternatives --install /usr/bin/javac javac /opt/java/bin/javac 300

update-alternatives --install /usr/bin/jar jar /opt/java/bin/jar 300

update-alternatives --install /usr/bin/javah javah /opt/java/bin/javah 300

update-alternatives --install /usr/bin/javap javap /opt/java/bin/javap 300

update-alternatives --config java

1. 安装其他必须的软件

Y

1. 安装ant

tar zxvf apache-ant-1.9.5-bin.tar.gz –C /opt/

vim /etc/profile

 export ANT\_HOME=/opt/apache-ant-1.9.5

export PATH=$PATH:$ANT\_HOME/bin

1. 安装findbugs

 tar zxvf findbugs-3.0.0.tar.gz –C /opt/

 vim /etc/profile

 export FINDBUGS\_HOME=/opt/findbugs-3.0.0

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

1. 安装protobuf

# tar zxvf protobuf-2.5.0.tar.gz(必须是2.5.0版本的，不然编译hadoop的时候报错)

# cd protobuf-2.5.0

# ./configure --prefix=/usr/local

# make && make install

1. 编译hadoop源码

# tar zxvf hadoop-2.6.0-src.tar.gz

# cd hadoop-2.6.0-src

# mvn package -Pdist,native,docs -DskipTests -Dtar

1. maven中央仓库的配置（增加访问速度，视具体情况选择）

vim /usr/share/maven/conf/setting.xml

<settings>

<localRepository>D:\maven\_new\_repository</localRepository>

</settings>

1. 配置hadoop

hadoop-2.6.0-x64.tar.gz

1、基础操作

tar -zxvf hadoop-2.6.0-x64.tar.gz -C /opt/

# chown -R hadoop:hadoop /opt/hadoop-2.6.0

# vi /etc/profile

 export HADOOP\_HOME=/opt/hadoop-2.6.0

 export PATH=$PATH:$HADOOP\_HOME/bin

# su hadoop

$ cd /opt/hadoop-2.6.0

$ mkdir -p dfs/name

$ mkdir -p dfs/data

$ mkdir -p tmp

$ cd etc/hadoop

2、配置所有slave节点

vim slaves

master

slave1

slave2

3、修改hadoop-env.sh和yarn-env.sh

export JAVA\_HOME=/opt/java

export HADOOP\_ROOT\_LOGGER=DEBUG,console

4、修改core-site.xml

<configuration>

<property>

<name>fs.defaultFS</name>

<value>hdfs://master:9000</value>

</property>

<property>

<name>io.file.buffer.size</name>

<value>131702</value>

</property>

<property>

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

<value>file:/opt/hadoop-2.6.0/tmp</value>

</property>

<!--

<property>

<name>hadoop.proxyuser.hadoop.hosts</name>

<value>\*</value>

</property>

<property>

<name>hadoop.proxyuser.hadoop.groups</name>

<value>\*</value>

</property>

<property>

<name>hadoop.native.lib</name>

<value>false</value>

</property>

-->

</configuration>

6、修改hdfs-site.xml

<configuration>

<property>

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

<value>file:/opt/hadoop-2.6.0/dfs/name</value>

</property>

<property>

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

<value>file:/opt/hadoop-2.6.0/dfs/data</value>

</property>

<property>

<name>dfs.replication</name>

<value>3</value>

</property>

<property>

<name>dfs.namenode.secondary.http-address</name>

<value>master:9001</value>

</property>

<property>

<name>dfs.webhdfs.enabled</name>

<value>true</value>

</property>

</configuration>

7、修改mapred-site.xml

<configuration>

<property>

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

<value>yarn</value>

</property>

<property>

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

<value>master:10020</value>

</property>

<property>

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

<value>master:19888</value>

</property>

</configuration>

8、修改yanr-site.xml

<configuration>

<!-- Site specific YARN configuration properties -->

<property>

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

<value>mapreduce\_shuffle</value>

</property>

<property>

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

<value>master</value>

</property>

<property>

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

<value>master:8032</value>

</property>

<property>

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

<value>master:8030</value>

</property>

<property>

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

<value>master:8031</value>

</property>

<property>

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

<value>master:8033</value>

</property>

<property>

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

<value>master:8088</value>

</property>

<!--

<property>

<name>yarn.nodemanager.resource.memory-mb</name>

<value>3000</value>

</property>

-->

</configuration>

9、格式化namenode

$HADOOP\_HOME/bin/hadoop namenode –format

只需要在master上执行一次就行

10、启动hdfs

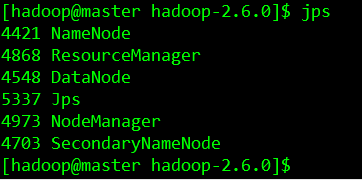
$HADOOP\_HOME/sbin/start-dfs.sh

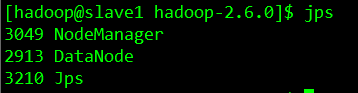
$HADOOP\_HOME/sbin/start-yarn.sh

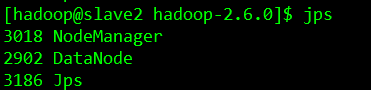
11、检查启动情况

http://192.168.1.7:8088

http://192.168.1.7:50070

master上

slave1和slave2上



12、问题及日志

$HADOOP\_HOME/logs

1. 安装spark

spark-1.4.1-bin-hadoop2.6.tgz

tar –zxvf spark-1.4.1-bin-hadoop2.6.tgz –C /opt/

cd /opt/ spark-1.4.1-bin-hadoop2.6/conf

cp slaves.template slaves

cp spark-env.sh.template spark-env.sh

vim spark-env.sh

加入 export SPARK\_HOME=/opt/ spark-1.4.1-bin-hadoop2.6

Export MASTER=spark://master:7077

Export SPARK\_MASTER\_IP=192.168.1.55:

Vim slaves

修改为 master

slave1

slave2

然后同步到各个节点

启动spark

$SPARK\_HOME/sbin/start-master.sh

$SPARK\_HOME/sbin/start-slaves.sh

或者 $SPARK\_HOME/sbin/start-all.sh

1. 参考资料

CentOS7-64bit 编译 Hadoop-2.5.0，并分布式安装

<http://my.oschina.net/u/1428349/blog/313646#OSC_h2_1>

1. 常见问题及补充
2. 在使用maven的时候，如果是jdk1.8，将会报错。
3. 在配置spark时，若各个节点的hostname与/etc/hosts文件中的映射名不一样的话，也会报错，这时，只要export MASTER= 以及export SPARK\_MASTER\_IP= 就行。
4. 各节点间的jdk以及Python版本需保持一致，否则会导致错误。
5. 报错: mkdir: Cannot create directory /user/tk. Name node is in safe mode.

使用命令：./hadoop dfsadmin -safemode leave

1. 性能调优
2. set(‘spark.executors.cores’ , ‘8’)

此处设置YARN管理时的cpu数量

1. 2016 01 24 13 58 20,
2. ,
3. 6864653,
4. 686465,
5. 1000236864653,
6. 222353027,
7. 38,
8. 28420,
9. 273,
10. -120,
11. -140,
12. ,
13. ,
14. ,
15. ,
16. ,
17. 1650,
18. 0,
19. ,
20. ,
21. ,
22. ,
23. 106.564454,
24. 29.572509,
25. ,
26. ,
27. ,
28. ,
29. ,
30. ,
31. ,
32. ,
33. ,
34. ,
35. ,
36. ,
37. ,
38. ,
39. ,
40. ,
41. ,
42. ,
43. ,
44. ,
45. ,
46. ,
47. 1650,
48. ,
49. ,
50. ,
51. ,
52. ,
53. ,
54. ,
55. ,
56. ,
57. ,
58. ,
59. ,
60. ,
61. ,
62. ,
63. ,
64. ,
65. ,
66. ,
67. ,
68. ,
69. ,
70. ,
71. ,
72. ,
73. ,
74. ,
75. ,
76. ,
77. ,
78. ,
79. ,
80. ,
81. ,
82. ,
83. ,
84. ,
85. ,
86. ,
87. ,
88. ,
89. ,
90. ,
91. ,
92. ,
93. ,
94. ,
95. ,
96. ,
97. ,
98. ,
99. ,
100. ,
101. ,
102. ,
103. ,
104. ,
105. ,
106. ,
107. ,
108. ,
109. ,
110. ,
111. ,
112. ,
113. ,
114. ,
115. ,
116. ,
117. ,
118. ,
119. ,
120. ,
121. ,
122. ,
123. ,
124. ,
125. ,
126. ,
127. ,
128. ,
129. ,
130. ,
131. ,
132. ,
133. ,
134. ,
135. ,
136. ,
137. ,
138. ,
139. ,
140. ,
141. ,
142. ,
143. ,
144. ,
145. ,
146. ,
147. ,
148. ,
149. 0,
150. 0,
151. 0,
152. 0,
153. 0,
154. ,
155. ,
156. ,
157. ,
158. ,
159. ,
160. ,
161. ,
162. ,
163. ,
164. ,
165. ,
166. ,
167. ,
168. ,

def getgridindexbase(lon, lat, xstep, ystep, LEN):

def getgridindex(lon, lat, tag):

return getgridindexbase(lon, lat, xstep1, ystep1, LEN1)

return getgridindexbase(lon, lat, xstep5, ystep5, LEN5)

(index, lon, lat) = getgridindex(line[16], line[17], tag)