目录

[hadoop集群安装步骤 1](#_Toc491877170)

[一、 系统环境 1](#_Toc491877171)

[二、 常用命令 2](#_Toc491877172)

[1. 查看系统版本： 2](#_Toc491877173)

[2. 启动集群 2](#_Toc491877174)

[3. 关闭集群 2](#_Toc491877175)

[4. 监控集群 2](#_Toc491877176)

[5. 单个进程启动/关闭 2](#_Toc491877177)

[三、 环境准备（所有服务器） 3](#_Toc491877178)

[6. 关闭防火墙并禁止开机自启动 3](#_Toc491877179)

[7. 设置主机名（其它类似） 3](#_Toc491877180)

[8. ssh免密登陆 3](#_Toc491877181)

[9. 修改hosts文件 3](#_Toc491877182)

[10. 修改文件句柄数 4](#_Toc491877183)

[11. 时间同步 4](#_Toc491877184)

[12. 上传安装包到node1服务器 4](#_Toc491877185)

[13. 安装JDK 5](#_Toc491877186)

[四、 安装zookeeper 6](#_Toc491877187)

[14. 解压zookeeper 6](#_Toc491877188)

[15. 配置zoo.cfg文件 6](#_Toc491877189)

[16. 添加myid，分发(安装个数为奇数) 7](#_Toc491877190)

[17. 分发： 7](#_Toc491877191)

[五、 安装Hadoop 8](#_Toc491877192)

[18. 解压hadoop 8](#_Toc491877193)

[19. 配置hadoop-env.sh 8](#_Toc491877194)

[20. 配置core-site.xml 8](#_Toc491877195)

[21. 配置yarn-site.xml 9](#_Toc491877196)

[22. 配置mapred-site.xml 13](#_Toc491877197)

[23. 配置hdfs-site.xml 13](#_Toc491877198)

[24. 配置capacity-scheduler.xml 15](#_Toc491877199)

[25. 配置slaves 18](#_Toc491877200)

[26. 修改$HADOOP\_HOME/sbin/hadoop-daemon.sh 18](#_Toc491877201)

[27. 修改$HADOOP\_HOME/sbin/yarn-daemon.sh 18](#_Toc491877202)

[28. 分发 19](#_Toc491877203)

[六、 安装hbase 20](#_Toc491877204)

[29. 解压hbase 20](#_Toc491877205)

[30. 修改$HBASE\_HOME/conf/hbase-env.sh,添加 20](#_Toc491877206)

[31. 修改regionservers文件 20](#_Toc491877207)

[32. 修改hbase-site.xml文件 20](#_Toc491877208)

[33. 分发 21](#_Toc491877209)

[七、 安装spark 22](#_Toc491877210)

[34. 解压spark 22](#_Toc491877211)

[35. 修改文件名：spark-2.0.2 22](#_Toc491877212)

[36. 配置spark-env.sh 22](#_Toc491877213)

[37. 配置spark-defaults.conf 22](#_Toc491877214)

[38. 配置slaves 23](#_Toc491877215)

[39. 分发 23](#_Toc491877216)

[八、 启动过程 24](#_Toc491877217)

[40. 同时开启所有zookeeper节点 24](#_Toc491877218)

[41. 启动所有journalnode节点 24](#_Toc491877219)

[42. 格式化namenode目录(主节点node1) 24](#_Toc491877220)

[43. 启动当前格式化的namenode进程(主节点node1) 24](#_Toc491877221)

[44. 在没有格式化的NN上 执行同步命令(副节点node2) 24](#_Toc491877222)

[45. 启动hdfs 24](#_Toc491877223)

[46. 启动yarn： 24](#_Toc491877224)

[47. 两台resourcemanager上启动resourcemanager 24](#_Toc491877225)

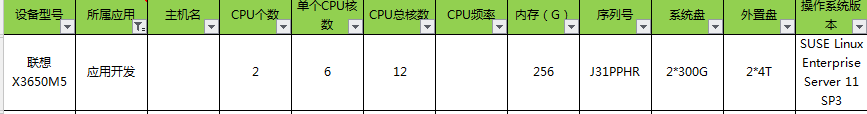
[48. 启动hbase 25](#_Toc491877226)

[49. 启动spark 25](#_Toc491877227)

[50. END 26](#_Toc491877228)

hadoop集群安装步骤

# 系统环境

****

**环境**

操作系统版本：SUSE Linux Enterprise Server 11 (x86\_64) SP3

主机名：

192.168.0.10 node1

192.168.0.11 node2

192.168.0.12 node3

192.168.0.13 node4

软件路径：/data/install

Hadoop集群路径：/data

JAVA\_HOME路径：/usr/jdk1.8.0\_66

**版本**

|  |  |  |
| --- | --- | --- |
| 组件名 | 版本 | 说明 |
| JRE | jdk-8u66-linux-x64.tar.gz |  |
| zookeeper | zookeeper-3.4.6.tar.gz |  |
| Hadoop | hadoop-2.7.3.tar.gz | 主程序包 |
| spark | spark-2.0.2-bin-hadoop2.7.tgz |  |
| hbase | hbase-1.2.5-bin.tar.gz |  |

# 常用命令

## 查看系统版本：

linux-n4ga:~ # uname –a #内核版本

Linux node1 3.0.76-0.11-default #1 SMP Fri Jun 14 08:21:43 UTC 2013 (ccab990) x86\_64 x86\_64 x86\_64 GNU/Linux

linux-n4ga:~ # lsb\_release #发行版本

LSB Version: core-2.0-noarch:core-3.2-noarch:core-4.0-noarch:core-2.0-x86\_64:core-3.2-x86\_64:core-4.0-x86\_64:desktop-4.0-amd64:desktop-4.0-noarch:graphics-2.0-amd64:graphics-2.0-noarch:graphics-3.2-amd64:graphics-3.2-noarch:graphics-4.0-amd64:graphics-4.0-noarch

linux-n4ga:~ # cat /etc/SuSE-release #补丁版本

SUSE Linux Enterprise Server 11 (x86\_64)

VERSION = 11

PATCHLEVEL = 3

node1:~ # cat /etc/issue

Welcome to SUSE Linux Enterprise Server 11 SP3 (x86\_64) - Kernel \r (\l).

node1:~ #

## 启动集群

start-dfs.sh

start-yarn.sh

## 关闭集群

stop-yarn.sh

stop-dfs.sh

## 监控集群

hdfs dfsadmin -report

## 单个进程启动/关闭

hadoop-daemon.sh start|stop namenode|datanode| journalnode

yarn-daemon.sh start |stop resourcemanager|nodemanager

<http://blog.chinaunix.net/uid-25723371-id-4943894.html>

# 环境准备（所有服务器）

## 关闭防火墙并禁止开机自启动

linux-n4ga:~ # rcSuSEfirewall2 stop

Shutting down the Firewall done

linux-n4ga:~ # chkconfig SuSEfirewall2\_setup off

linux-n4ga:~ # chkconfig SuSEfirewall2\_init off

linux-n4ga:~ # chkconfig --list|grep fire

SuSEfirewall2\_init 0:off 1:off 2:off 3:off 4:off 5:off 6:off

SuSEfirewall2\_setup 0:off 1:off 2:off 3:off 4:off 5:off 6:off

## 设置主机名（其它类似）

linux-n4ga:~ # hostname node1

linux-n4ga:~ # vim /etc/HOSTNAME

node1.site

## ssh免密登陆

node1:~ # ssh-keygen -t dsa -P '' -f ~/.ssh/id\_dsa

node1:~ # cat ~/.ssh/id\_dsa.pub >> ~/.ssh/authorized\_keys

node1:~ # ll -d .ssh/

drwx------ 2 root root 4096 Jun 5 08:50 .ssh/

node1:~ # ll .ssh/

total 12

-rw-r--r-- 1 root root 599 Jun 5 08:50 authorized\_keys

-rw------- 1 root root 672 Jun 5 08:50 id\_dsa

-rw-r--r-- 1 root root 599 Jun 5 08:50 id\_dsa.pub

把其它服务器的~/.ssh/id\_dsa.pub内容也追加到node1服务器的~/.ssh/authorized\_keys文件中，然后分发

scp –rp ~/.ssh/authorized\_keys root@192.168.0.11: ~/.ssh/

scp –rp ~/.ssh/authorized\_keys root@192.168.0.12: ~/.ssh/

scp –rp ~/.ssh/authorized\_keys root@192.168.0.13: ~/.ssh/

## 修改hosts文件

node1:~ # vim /etc/hosts

… …

ff02::2 ipv6-allrouters

ff02::3 ipv6-allhosts

192.168.0.10 node1

192.168.0.11 node2

192.168.0.12 node3

192.168.0.13 node4

分发：

scp -rp /etc/hosts [root@192.168.0.11:/etc/](mailto:root@192.168.0.11:/etc/)

scp -rp /etc/hosts [root@192.168.0.12:/etc/](mailto:root@192.168.0.12:/etc/)

scp -rp /etc/hosts root@192.168.0.13:/etc/

## 修改文件句柄数

node1:~ # vim /etc/security/limits.conf

\* soft nofile 24000

\* hard nofile 65535

\* soft nproc 24000

\* hard nproc 65535

node1:~ # source /etc/security/limits.conf

node1:~ # ulimit -n

24000

## 时间同步

测试（举例）

node1 :~ # /usr/sbin/ntpdate 192.168.0.10

13 Jun 13:49:41 ntpdate[8370]: adjust time server 192.168.0.10 offset -0.007294 sec

添加定时任务

node1 :~ # crontab –e

\*/10 \* \* \* \* /usr/sbin/ntpdate 192.168.0.10 > /dev/null 2>&1;/sbin/hwclock -w

node1:~ # service cron restart

Shutting down CRON daemon done

Starting CRON daemon done

node1:~ # date

Tue Jun 13 05:32:49 CST 2017

node1:~ #

## 上传安装包到node1服务器

node1:~ # mkdir –pv /data/install

node1:~ # cd /data/install

node1:~ # pwd

/data/install

上传安装包到/data/install目录下

node1:/data/install # ll

total 671968

-rw-r--r-- 1 root root 214092195 Jun 5 05:40 hadoop-2.7.3.tar.gz

-rw-r--r-- 1 root root 104584366 Jun 5 05:40 hbase-1.2.5-bin.tar.gz

-rw-r--r-- 1 root root 181287376 Jun 5 05:47 jdk-8u66-linux-x64.tar.gz

-rw-r--r-- 1 root root 187426587 Jun 5 05:40 spark-2.0.2-bin-hadoop2.7.tgz

-rw-r--r-- 1 root root 187426587 Jun 5 05:40 zookeeper-3.4.6.tar.gz

## 安装JDK

node1:~ # cd /data/install

node1:/data/install # tar -zxvf jdk-8u66-linux-x64.tar.gz -C /usr/

配置环境变量

node1:/data/install #vim /etc/profile

export JAVA\_HOME=/usr/jdk1.8.0\_66

export HADOOP\_HOME=/data/hadoop-2.7.3

export HBASE\_HOME=/data/hbase-1.2.5

export SPARK\_HOME=/data/spark-2.0.2

export ZOOKEEPER\_HOME=/data/zookeeper-3.4.6

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

export CLASSPATH=.:$JAVA\_HOME/lib:$JAVA\_HOME/jre/lib

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

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

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

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

node1:/opt # source /etc/profile

node1:~ # java –version #验证

java version "1.8.0\_66"

Java(TM) SE Runtime Environment (build 1.8.0\_66-b17)

Java HotSpot(TM) 64-Bit Server VM (build 25.66-b17, mixed mode)

node1:~ # echo $JAVA\_HOME

/usr/jdk1.8.0\_66

# 安装zookeeper

## 解压zookeeper

node1:~ # cd /data/install

node1:/data/install # tar -zxvf zookeeper-3.4.6.tar.gz -C /data/

## 配置zoo.cfg文件

node1:/data/install # cd /data/zookeeper-3.4.6/conf/ #进入conf目录

node1: /data/zookeeper-3.4.6/conf/ # cp zoo\_sample.cfg zoo.cfg #拷贝模板

node1: /data/zookeeper-3.4.6/conf/ # vi zoo.cfg

# The number of millinode2s of each tick

tickTime=2000

# The number of ticks that the initial

# synchronization phase can take

initLimit=10

# The number of ticks that can pass between

# sending a request and getting an acknowledgement

syncLimit=5

# the directory where the snapshot is stored.

# do not use /tmp for storage, /tmp here is just

# example sakes.

dataDir=/data/zookeeper-3.4.6/data

dataLogDir=/data/zookeeper-3.4.6/dataLog

# the port at which the clients will connect

clientPort=2181

# the maximum number of client connections.

# increase this if you need to handle more clients

#maxClientCnxns=60

#

# Be sure to read the maintenance section of the

# administrator guide before turning on autopurge.

#

# http://zookeeper.apache.org/doc/current/zookeeperAdmin.html#sc\_maintenance

#

# The number of snapshots to retain in dataDir

#autopurge.snapRetainCount=3

# Purge task interval in hours

# Set to "0" to disable auto purge feature

#autopurge.purgeInterval=1

server.1=node1:2888:3888

server.2=node2:2888:3888

server.3=node3:2888:3888

## 添加myid，分发(安装个数为奇数)

创建指定目录：dataDir目录下增加myid文件;myid中写当前zookeeper服务的id, 因为server.1=node1:2888:3888 server指定的是1,

node1: /data/zookeeper-3.4.6/conf/ # mkdir –pv /data/zookeeper-3.4.6/{data, dataLog}

node1: /data/zookeeper-3.4.6/conf/ # echo 1 > /data/zookeeper-3.4.6/data/myid

## 分发：

node1: /data/zookeeper-3.4.6/conf/ # scp -rp /data/zookeeper-3.4.6 root@192.168.0.11:/data

node1: /data/zookeeper-3.4.6/conf/ # scp -rp /data/zookeeper-3.4.6 root@192.168.0.12:/data

在其余机子配置，node2下面的myid是2，node3下面myid是3，这些都是根据server来的

node2: /data/zookeeper-3.4.6/conf/ # echo 2 > /data/zookeeper-3.4.6/data/myid

node3: /data/zookeeper-3.4.6/conf/ # echo 3> /data/zookeeper-3.4.6/data/myid

# 安装Hadoop

## 解压hadoop

node1:~ # cd /data/install

node1:/data/install # tar -zxvf hadoop-2.7.3.tar.gz -C /data/

## 配置hadoop-env.sh

node1:~ # vim /data/hadoop-2.7.3/etc/hadoop/hadoop-env.sh

export JAVA\_HOME=/usr/jdk1.8.0\_66

## 配置core-site.xml

node1:~ # vim /data/hadoop-2.7.3/etc/hadoop/core-site.xml

<?xml version="1.0" encoding="UTF-8"?>

<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<configuration>

<!--Yarn 需要使用 fs.defaultFS 指定NameNode URI -->

    <property>

        <name>fs.defaultFS</name>

        <value>hdfs://mycluster</value>

</property>

<property>

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

<value>/data/hadoop-2.7.3/data/tmp</value>

</property>

<property>

<name>ha.zookeeper.quorum</name>

<value>node1:2181,node2:2181,node3:2181</value>

<discription>zookeeper客户端连接地址</discription>

</property>

<property>

<name>ha.zookeeper.session-timeout.ms</name>

<value>10000</value>

</property>

<property>

<name>fs.trash.interval</name>

<value>1440</value>

<discription>以分钟为单位的垃圾回收时间，垃圾站中数据超过此时间，会被删除。如果是0，垃圾回收机制关闭。</discription>

</property>

<property>

<name>fs.trash.checkpoint.interval</name>

<value>1440</value>

<discription>以分钟为单位的垃圾回收检查间隔。</discription>

</property>

</configuration>

## 配置yarn-site.xml

node1:~ # vim /data/hadoop-2.7.3/etc/hadoop/yarn-site.xml #

<?xml version="1.0"?>

<configuration>

<property>

<name>yarn.app.mapreduce.am.scheduler.connection.wait.interval-ms</name>

<value>5000</value>

<discription>schelduler失联等待连接时间</discription>

</property>

<property>

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

<value>mapreduce\_shuffle</value>

<discription>NodeManager上运行的附属服务。需配置成mapreduce\_shuffle，才可运行MapReduce程序</discription>

</property>

<property>

<name>yarn.resourcemanager.ha.enabled</name>

<value>true</value>

<discription>是否启用RM HA，默认为false（不启用）</discription>

</property>

<property>

<name>yarn.resourcemanager.cluster-id</name>

<value>cluster1</value>

<discription>集群的Id，elector使用该值确保RM不会做为其它集群的active。</discription>

</property>

<property>

<name>yarn.resourcemanager.ha.rm-ids</name>

<value>rm1,rm2</value>

<discription>RMs的逻辑id列表,用逗号分隔,如:rm1,rm2 </discription>

</property>

<property>

<name>yarn.resourcemanager.hostname.rm1</name>

<value>node3</value>

<discription>RM的hostname</discription>

</property>

<property>

<name>yarn.resourcemanager.scheduler.address.rm1</name>

<value>${yarn.resourcemanager.hostname.rm1}:8030</value>

<discription>RM对AM暴露的地址,AM通过地址想RM申请资源,释放资源等</discription>

</property>

<property>

<name>yarn.resourcemanager.resource-tracker.address.rm1</name>

<value>${yarn.resourcemanager.hostname.rm1}:8031</value>

<discription>RM对NM暴露地址,NM通过该地址向RM汇报心跳,领取任务等</discription>

</property>

<property>

<name>yarn.resourcemanager.address.rm1</name>

<value>${yarn.resourcemanager.hostname.rm1}:8032</value>

<discription>RM对客户端暴露的地址,客户端通过该地址向RM提交应用程序等</discription>

</property>

<property>

<name>yarn.resourcemanager.admin.address.rm1</name>

<value>${yarn.resourcemanager.hostname.rm1}:8033</value>

<discription>RM对管理员暴露的地址.管理员通过该地址向RM发送管理命令等</discription>

</property>

<property>

<name>yarn.resourcemanager.webapp.address.rm1</name>

<value>${yarn.resourcemanager.hostname.rm1}:8088</value>

<discription>RM对外暴露的web http地址，用户可通过该地址在浏览器中查看集群信息</discription>

</property>

<property>

<name>yarn.resourcemanager.hostname.rm2</name>

<value>node4</value>

</property>

<property>

<name>yarn.resourcemanager.scheduler.address.rm2</name>

<value>${yarn.resourcemanager.hostname.rm2}:8030</value>

</property>

<property>

<name>yarn.resourcemanager.resource-tracker.address.rm2</name>

<value>${yarn.resourcemanager.hostname.rm2}:8031</value>

</property>

<property>

<name>yarn.resourcemanager.address.rm2</name>

<value>${yarn.resourcemanager.hostname.rm2}:8032</value>

</property>

<property>

<name>yarn.resourcemanager.admin.address.rm2</name>

<value>${yarn.resourcemanager.hostname.rm2}:8033</value>

</property>

<property>

<name>yarn.resourcemanager.webapp.address.rm2</name>

<value>${yarn.resourcemanager.hostname.rm2}:8088</value>

</property>

<property>

<name>yarn.resourcemanager.recovery.enabled</name>

<value>true</value>

<discription>默认值为false，也就是说resourcemanager挂了相应的正在运行的任务在rm恢复后不能重新启动</discription>

</property>

<property>

<name>yarn.resourcemanager.store.class</name>

<value>org.apache.hadoop.yarn.server.resourcemanager.recovery.ZKRMStateStore</value>

<discription>状态存储的类</discription>

</property>

<property>

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

<value>node1:2181,node2:2181,node3:2181</value>

</property>

<property>

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

<value> 240000</value>

<discription>该节点上nodemanager可使用的物理内存总量</discription>

</property>

<property>

<name>yarn.nodemanager.resource.cpu-vcores</name>

<value>24</value>

<discription>该节点上nodemanager可使用的虚拟CPU个数</discription>

</property>

<property>

<name>yarn.scheduler.minimum-allocation-mb</name>

<value>1024</value>

<discription>单个任务可申请的最小物理内存量</discription>

</property>

<property>

<name>yarn.scheduler.maximum-allocation-mb</name>

<value>240000</value>

<discription>单个任务可申请的最大物理内存量</discription>

</property>

<property>

<name>yarn.scheduler.minimum-allocation-vcores</name>

<value>1</value>

<discription>单个任务可申请的最小虚拟CPU个数</discription>

</property>

<property>

<name>yarn.scheduler.maximum-allocation-vcores</name>

<value>24</value>

<discription>单个任务可申请的最大虚拟CPU个数</discription>

</property>

<property>

<name>yarn.nodemanager.vmem-pmem-ratio</name>

<value>4</value>

<discription>任务每使用1MB物理内存，最多可使用虚拟内存量，默认是2.1。</discription>

</property>

</configuration>

## 配置mapred-site.xml

node1:~ # cp /data/hadoop-2.7.3/etc/hadoop/mapred-site.xml{.template,}

node1:~ # vim /data/hadoop-2.7.3/etc/hadoop/mapred-site.xml

<?xml version="1.0"?>

<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<configuration>

<property>

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

<value>yarn</value>

</property>

</configuration>

## 配置hdfs-site.xml

node1:~ # vim /data/hadoop-2.7.3/etc/hadoop/hdfs-site.xml

<?xml version="1.0" encoding="UTF-8"?>

<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<configuration>

<property>

<name>dfs.replication</name>

<value>2</value>

<description>保存副本数</description>

</property>

<property>

<name>dfs.nameservices</name>

<value>mycluster</value>

</property>

<property>

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

<value>nn1,nn2</value>

</property>

<property>

<name>dfs.namenode.rpc-address.mycluster.nn1</name>

<value>node1:8020</value>

</property>

<property>

<name>dfs.namenode.rpc-address.mycluster.nn2</name>

<value>node2:8020</value>

</property>

<property>

<name>dfs.namenode.http-address.mycluster.nn1</name>

<value>node1:50070</value>

</property>

<property>

<name>dfs.namenode.http-address.mycluster.nn2</name>

<value>node2:50070</value>

</property>

<property>

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

<value>qjournal://node1:8485;node2:8485;node3:8485/mycluster</value>

</property>

<property>

<name>dfs.client.failover.proxy.provider.mycluster</name>

<value>org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider</value>

</property>

<property>

<name>dfs.ha.fencing.methods</name>

<value>sshfence</value>

</property>

<property>

<name>dfs.ha.fencing.ssh.private-key-files</name>

<value>/root/.ssh/id\_dsa</value>

</property>

<property>

<name>dfs.journalnode.edits.dir</name>

<value>/data/ hadoop-2.7.3/data/journal</value>

</property>

<property>

<name>dfs.permissions.superusergroup</name>

<value>root</value>

<description>超级用户组名</description>

</property>

<property>

<name>dfs.ha.automatic-failover.enabled</name>

<value>true</value>

<description>开启自动故障转移</description>

</property>

</configuration>

新建相应目录

node1:~ # mkdir -pv /data/ hadoop-2.7.3/data/{journal,tmp}

## 配置capacity-scheduler.xml

<configuration>

<property>

<name>yarn.scheduler.capacity.maximum-applications</name>

<value>10000</value>

<description>

Maximum number of applications that can be pending and running.

</description>

</property>

<property>

<name>yarn.scheduler.capacity.maximum-am-resource-percent</name>

<value>0.1</value>

<description>

Maximum percent of resources in the cluster which can be used to run

application masters i.e. controls number of concurrent running

applications.

</description>

</property>

<property>

<name>yarn.scheduler.capacity.resource-calculator</name>

<value>org.apache.hadoop.yarn.util.resource.DominantResourceCalculator</value>

<description>

The ResourceCalculator implementation to be used to compare

Resources in the scheduler.

The default i.e. DefaultResourceCalculator only uses Memory while

DominantResourceCalculator uses dominant-resource to compare

multi-dimensional resources such as Memory, CPU etc.

</description>

</property>

<property>

<name>yarn.scheduler.capacity.root.queues</name>

<value>default</value>

<description>

The queues at the this level (root is the root queue).

</description>

</property>

<property>

<name>yarn.scheduler.capacity.root.default.capacity</name>

<value>100</value>

<description>Default queue target capacity.</description>

</property>

<property>

<name>yarn.scheduler.capacity.root.default.user-limit-factor</name>

<value>1</value>

<description>

Default queue user limit a percentage from 0.0 to 1.0.

</description>

</property>

<property>

<name>yarn.scheduler.capacity.root.default.maximum-capacity</name>

<value>100</value>

<description>

The maximum capacity of the default queue.

</description>

</property>

<property>

<name>yarn.scheduler.capacity.root.default.state</name>

<value>RUNNING</value>

<description>

The state of the default queue. State can be one of RUNNING or STOPPED.

</description>

</property>

<property>

<name>yarn.scheduler.capacity.root.default.acl\_submit\_applications</name>

<value>\*</value>

<description>

The ACL of who can submit jobs to the default queue.

</description>

</property>

<property>

<name>yarn.scheduler.capacity.root.default.acl\_administer\_queue</name>

<value>\*</value>

<description>

The ACL of who can administer jobs on the default queue.

</description>

</property>

<property>

<name>yarn.scheduler.capacity.node-locality-delay</name>

<value>40</value>

<description>

Number of missed scheduling opportunities after which the CapacityScheduler

attempts to schedule rack-local containers.

Typically this should be set to number of nodes in the cluster, By default is setting

approximately number of nodes in one rack which is 40.

</description>

</property>

<property>

<name>yarn.scheduler.capacity.queue-mappings</name>

<value></value>

<description>

A list of mappings that will be used to assign jobs to queues

The syntax for this list is [u|g]:[name]:[queue\_name][,next mapping]\*

Typically this list will be used to map users to queues,

for example, u:%user:%user maps all users to queues with the same name

as the user.

</description>

</property>

<property>

<name>yarn.scheduler.capacity.queue-mappings-override.enable</name>

<value>false</value>

<description>

If a queue mapping is present, will it override the value specified

by the user? This can be used by administrators to place jobs in queues

that are different than the one specified by the user.

The default is false.

</description>

</property>

</configuration>

## 配置slaves

node1:~ # vim /data/hadoop-2.7.3/etc/hadoop/

node1

node2

node3

node4

## 修改$HADOOP\_HOME/sbin/hadoop-daemon.sh

node1: /data/hadoop-2.7.3 # cd /data/hadoop-2.7.3/sbin/

#添加：

node1: /data/hadoop-2.7.3/sbin # HADOOP\_PID\_DIR=/data/hdfs/pids

## 修改$HADOOP\_HOME/sbin/yarn-daemon.sh

#添加：

node1: /data/hadoop-2.7.3/sbin # HADOOP\_PID\_DIR=/data/hdfs/pids

## 分发

node1: /data/hadoop-2.7.3/etc/hadoop/ # scp -rp /data/hadoop-2.7.3 root@192.168.0.11:/data

node1: /data/hadoop-2.7.3/etc/hadoop/ # scp -rp /data/hadoop-2.7.3 root@192.168.0.12:/data

node1: /data/hadoop-2.7.3/etc/hadoop/ # scp -rp /data/hadoop-2.7.3 root@192.168.0.13:/data

# 安装hbase

## 解压hbase

node1:/data # cd /data/install

node1:/data/install # tar -zxvf hbase-1.2.5-bin.tar.gz -C /data

## 修改$HBASE\_HOME/conf/hbase-env.sh,添加

node1:/data # cd /data/hbase-1.2.5/conf

node1: /data/hbase-1.2.5 # vim hbase-env.sh

export HBASE\_HOME=/data/hbase-1.2.5

export JAVA\_HOME=/usr/jdk1.8.0\_66

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

export HBASE\_LIBRARY\_PATH=$HBASE\_LIBRARY\_PATH:$HBASE\_HOME/lib/native/

#设置到Hadoop的etc/hadoop目录是用来引导Hbase找到Hadoop,也就是说hbase和hadoop进行关联【必须设置,否则hmaster起不来】

export HBASE\_CLASSPATH=$HADOOP\_HOME/etc/hadoop

export HBASE\_MANAGES\_ZK=false #不启用hbase自带的zookeeper

export HBASE\_PID\_DIR=/data/hdfs/pids

export HBASE\_SSH\_OPTS="-o ConnectTimeout=1 -p 36928" #ssh端口；

## 修改regionservers文件

node1: /data/hbase-1.2.5 # vim regionservers

node1

node2

node3

node4

node1: /data/hbase-1.2.5 #

## 修改hbase-site.xml文件

node1:/data/hbase-1.2.5/conf # vim hbase-site.xml

<configuration>

<property>

<name>hbase.rootdir</name>

<value>hdfs://mycluster/hbase</value>

</property>

<property>

     <name>hbase.zookeeper.quorum</name>

     <value>node1,node2,node3</value>

</property>

<property>

     <name>hbase.zookeeper.property.clientPort</name>

     <value>2181</value>

   </property>

</configuration>

## 分发

node1: /data/hbase-1.2.5/conf # scp -rp /data/hbase-1.2.5 root@192.168.0.11:/data

node1: /data/hbase-1.2.5/conf # scp -rp /data/hbase-1.2.5 root@192.168.0.12:/data

node1: /data/hbase-1.2.5/conf # scp -rp /data/hbase-1.2.5 root@192.168.0.13:/data

# 安装spark

## 解压spark

node1:/data #cd /data/install

node1:/data/install # tar -zxvf spark-2.0.2-bin-hadoop2.7.tgz -C /data

## 修改文件名：spark-2.0.2

node1:/data # mv spark-2.0.2-bin-hadoop2.7 spark-2.0.2

## 配置spark-env.sh

node1:/data #cd /data/spark-2.0.2/conf/

node1: /data/spark-2.0.2/conf/ #cp spark-env.sh.template spark-env.sh

node1: /data/spark-2.0.2/conf/ #vim spark-env.sh

#添加：

export JAVA\_HOME=/usr/jdk1.8.0\_66

export SPARK\_PID\_DIR=/data/ spark-2.0.2/conf/pids

#设置内存

export SPARK\_WORKER\_MEMORY=240g

export HADOOP\_CONF\_DIR=$HADOOP\_HOME/etc/hadoop

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

export SPARK\_MASTER\_PORT=7077

export SPARK\_WORKER\_INSTANCES=1

export SPARK\_HISTORY\_OPTS="-Dspark.history.ui.port=18080 -Dspark.history.retainedApplications=3 -Dspark.history.fs.logDirectory=hdfs://mycluster/directory"

# 限制程序申请资源最大核数

export SPARK\_MASTER\_OPTS="-Dspark.deploy.defaultCores=12"

export SPARK\_SSH\_OPTS="-p 36928 -o StrictHostKeyChecking=no $SPARK\_SSH\_OPTS"

export SPARK\_HISTORY\_OPTS="-Dspark.history.ui.port=18080 -Dspark.history.retainedApplications=3 -Dspark.history.fs.logDirectory=hdfs://mycluster/directory"

#内存小于32G，配下面的

export SPARK\_JAVA\_OPTS="-XX:+UseCompressedOops -XX:+UseCompressedStrings $SPARK\_JAVA\_OPTS"

## 配置spark-defaults.conf

node1:/data #cd /data/spark-2.0.2/conf/

node1: /data/spark-2.0.2/conf/ #cp spark-defaults.conf.template spark-defaults.conf

node1: /data/spark-2.0.2/conf/ #vi spark-defaults.conf

#添加

spark.serializer org.apache.spark.serializer.KryoSerializer

spark.eventLog.enabled true

spark.eventLog.dir hdfs://mycluster/directory

spark.local.dir /data/spark-2.0.2/sparktmp

## 配置slaves

node1:/data #cd /data/spark-2.0.2/conf/

node1: /data/spark-2.0.2/conf/ #mv slaves.template slaves

node1: /data/spark-2.0.2/conf/ # vim slaves

node1

node2

node3

node4

node1: /data/spark-2.0.2/conf/ #

## 分发

node1: /data/spark-2.0.2/conf/ # scp -rp /data/spark-2.0.2 root@192.168.0.11:/data

node1: /data/spark-2.0.2/conf/ # scp -rp /data/spark-2.0.2 root@192.168.0.12:/data

node1: /data/spark-2.0.2/conf/ # scp -rp /data/spark-2.0.2 root@192.168.0.13:/data

# 启动过程

## 同时开启所有zookeeper节点

node1:/data #cd /data/zookeeper-3.4.6/bin

node1: /data/zookeeper-3.4.6/bin #zkServer.sh start

node2: /data/zookeeper-3.4.6/bin #zkServer.sh start

node3: /data/zookeeper-3.4.6/bin #zkServer.sh start

## 启动所有journalnode节点

node1:/data #cd /data/hadoop-2.7.3

node1:/data/hadoop-2.7.3 #sbin/hadoop-daemon.sh start journalnode

node2:/data/hadoop-2.7.3 #sbin/hadoop-daemon.sh start journalnode

node3:/data/hadoop-2.7.3 #sbin/hadoop-daemon.sh start journalnode

## 格式化namenode目录(主节点node1)

node1:/data #cd /data/hadoop-2.7.3

node1:/data/hadoop-2.7.3 #./bin/hdfs namenode -format

## 启动当前格式化的namenode进程(主节点node1)

node1:/data/hadoop-2.7.3 #./sbin/hadoop-daemon.sh start namenode

## 在没有格式化的NN上 执行同步命令(副节点node2)

node2:/data/hadoop-2.7.3 #./bin/hdfs namenode -bootstrapStandby

## 启动hdfs

node1:/data/hadoop-2.7.3 #./sbin/hadoop-daemon.sh start namenode

node1:/data/hadoop-2.7.3 #./sbin/start-dfs.sh

## 启动yarn：

node1:~ # $HADOOP\_HOME/sbin/start-yarn.sh

## 两台resourcemanager上启动resourcemanager

node3:~ # $HADOOP\_HOME/sbin/yarn-daemon.sh start resourcemanager

node4:~ # $HADOOP\_HOME/sbin/yarn-daemon.sh start resourcemanager

HDFS和yarn的web控制台默认监听端口分别为50070和8088。可以通过浏览放访问查看运行情况。

停止命令：

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

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

如果一切正常，使用jps可以查看到正在运行的Hadoop服务，在我机器上的显示结果为：

7312 Jps

1793 NameNode

2163 JournalNode

357 NodeManager

2696 QuorumPeerMain

14428 DFSZKFailoverController

1917 DataNode

## 启动hbase

node1:/data/hadoop-2.7.3 #cd /data/hbase-1.2.5/bin

node1:/data/hbase-1.2.5/bin #./start-hbase.sh

node1:/data/hbase-1.2.5/bin # jps

7312 Jps

8463 HMaster

1793 NameNode

2163 JournalNode

357 NodeManager

14632 HRegionServer

2696 QuorumPeerMain

14428 DFSZKFailoverController

1917 DataNode

Hbase web页面<http://node1:16010>

## 启动spark

node1: /data/hbase-1.2.5/bin #cd /data /spark-2.0.2/sbin

node1: /data /spark-2.0.2/sbin #./start-all.sh

node1: /data /spark-2.0.2/sbin #./start-history-server.sh

node1:/data/spark-2.0.2/sbin # jps

7312 Jps

8463 HMaster

1793 NameNode

2163 JournalNode

4901 Worker

357 NodeManager

14632 HRegionServer

2696 QuorumPeerMain

14428 DFSZKFailoverController

1917 DataNode

1722 Master

node1:/data/spark-2.0.2/sbin #

spark的master web页面访问 <http://node1:8080>

spark的app历史日志页面访问 <http://node1:18080>

## END