# 第1章：物理机环境准备

**1）修改主机名**

（1）修改主机名称

[root@hadoop102 ~]# hostnamectl --static set-hostname hadoop102

（2）配置主机名称映射，打开/etc/hosts

[root@hadoop102 ~]# vim /etc/hosts

添加如下内容

172.31.237.2 hadoop102

172.31.237.3 hadoop103

172.31.237.4 hadoop104

172.31.237.5 hadoop105

172.31.237.6 hadoop106

172.31.237.7 hadoop107

172.31.237.8 hadoop108

172.31.237.9 hadoop109

（3）修改window10的主机映射文件（hosts文件）

（a）进入C:\Windows\System32\drivers\etc路径

（b）拷贝hosts文件到桌面

（c）打开桌面hosts文件并添加如下内容

172.31.237.2 hadoop102

172.31.237.3 hadoop103

172.31.237.4 hadoop104

172.31.237.5 hadoop105

172.31.237.6 hadoop106

172.31.237.7 hadoop107

172.31.237.8 hadoop108

172.31.237.9 hadoop109

（d）将桌面hosts文件覆盖C:\Windows\System32\drivers\etc路径hosts文件

**2）关闭防火墙**

[root@hadoop102 ~]# systemctl stop firewalld

[root@hadoop102 ~]# systemctl disable firewalld

***3）配置普通用户（xiao）具有root权限***

[root@hadoop102 ~]# vim /etc/sudoers

修改/etc/sudoers文件，找到下面一行（102行），在%wheel下面添加一行：

## Allow root to run any commands anywhere

root ALL=(ALL) ALL

%wheel ALL=(ALL) ALL

xiao ALL=(ALL) NOPASSWD: ALL

**4）在/opt目录下创建文件夹**

（1）在/opt目录下创建module、software文件夹

[root@hadoop102 opt]# mkdir /opt/module /opt/software

（2）修改module、software文件夹的所有者

[root@hadoop102 opt]# chown xiao:xiao /opt/module /opt/software

**5）重启**

[root@hadoop102 module]# reboot

# 第2章：集群分发脚本编写

## 2.1 编写集群分发脚本xsync

**1）scp（secure copy）安全拷贝**

（1）scp定义：

scp可以实现服务器与服务器之间的数据拷贝。（from server1 to server2）

（2）基本语法

scp -r $pdir/$fname $user@hadoop$host:$pdir/$fname

命令 递归 要拷贝的文件路径/名称 目的用户@主机:目的路径/名称

（3）案例实操

（a）在hadoop102上，将hadoop102中/opt/module目录下的软件拷贝到hadoop103上。

[xiao@hadoop102 ~]$ scp -r /opt/module/\* xiao@hadoop103:/opt/module

（b）在hadoop104上，将hadoop102服务器上的/opt/module目录下的软件拷贝到hadoop104上。

[xiao@hadoop104 opt]$ scp -r xiao@hadoop102:/opt/module/\* xiao@hadoop104:/opt/module

注意：拷贝过来的/opt/module目录，别忘了在hadoop102、hadoop103、hadoop104上修改所有文件的，所有者和所有者组。sudo chown xiao:xiao -R /opt/module

**2）rsync远程同步工具**

rsync主要用于备份和镜像。具有速度快、避免复制相同内容和支持符号链接的优点。

rsync和scp区别：用rsync做文件的复制要比scp的速度快，rsync只对差异文件做更新。scp是把所有文件都复制过去。

（1）基本语法

rsync -av $pdir/$fname $user@hadoop$host:$pdir/$fname

命令 选项参数 要拷贝的文件路径/名称 目的用户@主机:目的路径/名称

选项参数说明

|  |  |
| --- | --- |
| 选项 | 功能 |
| -a | 归档拷贝 |
| -v | 显示复制过程 |

（2）案例实操

（a）将hadoop102中/etc/profile.d/my\_env.sh文件拷贝到hadoop103的/etc/profile.d/my\_env.sh上。

[xiao@hadoop102 ~]$ sudo rsync -av /etc/profile.d/my\_env.sh root@hadoop103:/etc/profile.d/my\_env.sh

（b）将hadoop102中/etc/profile.d/my\_env.sh文件拷贝到hadoop103的/etc/profile.d/my\_env.sh上。

[xiao@hadoop102 ~]$ sudo rsync -av /etc/profile.d/my\_env.sh root@hadoop104:/etc/profile.d/my\_env.sh

注意：拷贝过来的配置文件别忘了source一下/etc/profile.d/my\_env.sh。

**3）xsync集群分发脚本**

（1）需求：循环复制文件到所有节点的相同目录下

（2）需求分析：

（a）rsync命令原始拷贝：

rsync -av /opt/module root@hadoop103:/opt/

（b）期望脚本：

xsync要同步的文件名称

（c）说明：在/home/xiao/bin这个目录下存放的脚本，xiao用户可以在系统任何地方直接执行。

（3）脚本实现

（a）在用的家目录/home/xiao下创建bin文件夹

[xiao@hadoop102 ~]$ mkdir bin

（b）在/home/xiao/bin目录下创建xsync文件，以便全局调用

[xiao@hadoop102 ~]$ cd /home/xiao/bin

[xiao@hadoop102 ~]$ vim xsync

在该文件中编写如下代码

#!/bin/bash

#1. 判断参数个数

if [ $# -lt 1 ]

then

echo Not Enough Arguement!

exit;

fi

#2. 遍历集群所有机器

for host in hadoop102 hadoop103 hadoop104

do

echo ==================== $host ====================

#3. 遍历所有目录，挨个发送

for file in $@

do

#4 判断文件是否存在

if [ -e $file ]

then

#5. 获取父目录

pdir=$(cd -P $(dirname $file); pwd)

#6. 获取当前文件的名称

fname=$(basename $file)

ssh $host "mkdir -p $pdir"

rsync -av $pdir/$fname $host:$pdir

else

echo $file does not exists!

fi

done

done

（c）修改脚本 xsync 具有执行权限

[xiao@hadoop102 bin]$ chmod +x xsync

（d）测试脚本

[xiao@hadoop102 bin]$ xsync xsync

## 2.2 SSH无密登录配置

**1）配置ssh**

（1）基本语法

ssh另一台电脑的IP地址

（2）ssh连接时出现Host key verification failed的解决方法

[xiao@hadoop102 ~]$ ssh hadoop103

出现：

The authenticity of host '192.168.1.103 (192.168.1.103)' can't be established.

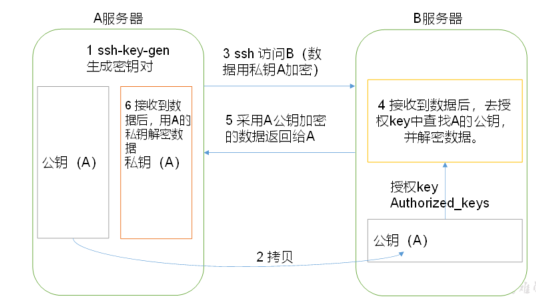
RSA key fingerprint is cf:1e:de:d7:d0:4c:2d:98:60:b4:fd:ae:b1:2d:ad:06.

Are you sure you want to continue connecting (yes/no)?

（3）解决方案如下：直接输入yes

**2）无密钥配置**

（1）免密登录原理



（2）生成公钥和私钥：

[xiao@hadoop102 .ssh]$ ssh-keygen -t rsa

然后敲（三个回车），就会生成两个文件id\_rsa（私钥）、id\_rsa.pub（公钥）

（3）将公钥拷贝到要免密登录的目标机器上

[xiao@hadoop102 .ssh]$ ssh-copy-id hadoop102

[xiao@hadoop102 .ssh]$ ssh-copy-id hadoop103

[xiao@hadoop102 .ssh]$ ssh-copy-id hadoop104

注意：

还需要在hadoop102上采用root账号，配置一下无密登录到hadoop102、hadoop103、hadoop104；

还需要在hadoop103上采用xiao账号配置一下无密登录到hadoop102、hadoop103、hadoop104服务器上。

**3）.ssh文件夹下（~/.ssh）的文件功能解释**

|  |  |
| --- | --- |
| known\_hosts | 记录ssh访问过计算机的公钥(public key) |
| id\_rsa | 生成的私钥 |
| id\_rsa.pub | 生成的公钥 |
| authorized\_keys | 存放授权过的无密登录服务器公钥 |

# 第3章：JDK安装

**1）卸载现有JDK（3台节点）**

[xiao@hadoop102 opt]# sudo rpm -qa | grep -i java | xargs -n1 sudo rpm -e --nodeps

[xiao@hadoop103 opt]# sudo rpm -qa | grep -i java | xargs -n1 sudo rpm -e --nodeps

[xiao@hadoop104 opt]# sudo rpm -qa | grep -i java | xargs -n1 sudo rpm -e --nodeps

**2）用Xftp5工具将JDK导入到hadoop102的/opt/software文件夹下面**

**3）在Linux系统下的opt目录中查看软件包是否导入成功**

[xiao@hadoop102 software]# ls /opt/software/

看到如下结果：

jdk-8u212-linux-x64.tar.gz

**4）解压JDK到/opt/module目录下**

[xiao@hadoop102 software]# tar -zxvf jdk-8u212-linux-x64.tar.gz -C /opt/module/

**5）配置JDK环境变量**

（1）新建/etc/profile.d/my\_env.sh文件

[xiao@hadoop102 module]# sudo vim /etc/profile.d/my\_env.sh

添加如下内容，然后保存（:wq）退出

#JAVA\_HOME

export JAVA\_HOME=/opt/module/jdk1.8.0\_212

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

（2）让环境变量生效

[xiao@hadoop102 software]$ source /etc/profile.d/my\_env.sh

**6）测试JDK是否安装成功**

[xiao@hadoop102 module]# java -version

如果能看到以下结果、则Java正常安装

java version "1.8.0\_212"

**7）分发JDK**

[xiao@hadoop102 module]$ xsync /opt/module/jdk1.8.0\_212/

**8）分发环境变量配置文件**

[xiao@hadoop102 module]$ sudo /home/xiao/bin/xsync /etc/profile.d/my\_env.sh

**9）分别在hadoop103、hadoop104上执行source**

[xiao@hadoop103 module]$ source /etc/profile.d/my\_env.sh

[xiao@hadoop104 module]$ source /etc/profile.d/my\_env.sh

# 第4章：Hadoop安装

## 1.1 搭建集群

**1）集群部署规划**

注意：NameNode和SecondaryNameNode不要安装在同一台服务器

注意：ResourceManager也很消耗内存，不要和NameNode、SecondaryNameNode配置在同一台机器上。

|  |  |  |  |
| --- | --- | --- | --- |
|  | hadoop102 | hadoop103 | hadoop104 |
| HDFS | NameNode  DataNode | DataNode | SecondaryNameNode  DataNode |
| YARN | NodeManager | ResourceManager  NodeManager | NodeManager |

**2）用Xftp5工具将hadoop-3.1.3.tar.gz导入到opt目录下面的software文件夹下面**

**3）进入到Hadoop安装包路径下**

[xiao@hadoop102 ~]$ cd /opt/software/

**4）解压安装文件到/opt/module下面**

[xiao@hadoop102 software]$ tar -zxvf hadoop-3.1.3.tar.gz -C /opt/module/

**5）查看是否解压成功**

[xiao@hadoop102 software]$ ls /opt/module/

hadoop-3.1.3

**6）将Hadoop添加到环境变量**

（1）获取Hadoop安装路径

[xiao@hadoop102 hadoop-3.1.3]$ pwd

/opt/module/hadoop-3.1.3

（2）打开/etc/profile.d/my\_env.sh文件

[xiao@hadoop102 hadoop-3.1.3]$ sudo vim /etc/profile.d/my\_env.sh

添加如下内容

##HADOOP\_HOME

export HADOOP\_HOME=/opt/module/hadoop-3.1.3

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

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

（3）保存后退出

:wq

（4）分发环境变量文件

[xiao@hadoop102 hadoop-3.1.3]$ sudo /home/xiao/bin/xsync /etc/profile.d/my\_env.sh

（5）source 是之生效（3台节点）

[xiao@hadoop102 module]$ source /etc/profile.d/my\_env.sh

[xiao@hadoop103 module]$ source /etc/profile.d/my\_env.sh

[xiao@hadoop104 module]$ source /etc/profile.d/my\_env.sh

## 1.2 配置集群

1）核心配置文件

配置core-site.xml

[xiao@hadoop102 ~]$ cd $HADOOP\_HOME/etc/hadoop

[xiao@hadoop102 hadoop]$ vim core-site.xml

文件内容如下：

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

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

<configuration>

<!-- 指定NameNode的地址 -->

<property>

<name>fs.defaultFS</name>

<value>hdfs://hadoop102:8020</value>

</property>

<!-- 指定hadoop数据的存储目录 -->

<property>

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

<value>/opt/module/hadoop-3.1.3/data</value>

</property>

<!-- 配置HDFS网页登录使用的静态用户为xiao -->

<property>

<name>hadoop.http.staticuser.user</name>

<value>xiao</value>

</property>

<!-- 配置该xiao(superUser)允许通过代理访问的主机节点 -->

<property>

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

<value>\*</value>

</property>

<!-- 配置该xiao(superUser)允许通过代理用户所属组 -->

<property>

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

<value>\*</value>

</property>

<!-- 配置该xiao(superUser)允许通过代理的用户-->

<property>

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

<value>\*</value>

</property>

</configuration>

2）HDFS配置文件

配置hdfs-site.xml

[xiao@hadoop102 hadoop]$ vim hdfs-site.xml

文件内容如下：

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

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

<configuration>

<!-- nn web端访问地址-->

<property>

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

<value>hadoop102:9870</value>

</property>

<!-- 2nn web端访问地址-->

<property>

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

<value>hadoop104:9868</value>

</property>

<!-- 测试环境指定HDFS副本的数量3 -->

<property>

<name>dfs.replication</name>

<value>3</value>

</property>

</configuration>

3）YARN配置文件

配置yarn-site.xml

[xiao@hadoop102 hadoop]$ vim yarn-site.xml

文件内容如下：

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

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

<configuration>

<!-- 指定MR走shuffle -->

<property>

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

<value>mapreduce\_shuffle</value>

</property>

<!-- 指定ResourceManager的地址-->

<property>

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

<value>hadoop103</value>

</property>

<!-- task继承nodemanager环境变量-->

<property>

<name>yarn.nodemanager.env-whitelist</name>

<value>JAVA\_HOME,HADOOP\_COMMON\_HOME,HADOOP\_HDFS\_HOME,HADOOP\_CONF\_DIR,CLASSPATH\_PREPEND\_DISTCACHE,HADOOP\_YARN\_HOME,HADOOP\_MAPRED\_HOME</value>

</property>

<!-- yarn容器允许分配的最大最小内存 -->

<property>

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

<value>10240</value>

</property>

<property>

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

<value>102400</value>

</property>

<!-- yarn容器允许管理的物理内存大小 -->

<property>

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

<value>102400</value>

</property>

<!-- 关闭yarn对物理内存和虚拟内存的限制检查 -->

<property>

<name>yarn.nodemanager.pmem-check-enabled</name>

<value>false</value>

</property>

<property>

<name>yarn.nodemanager.vmem-check-enabled</name>

<value>false</value>

</property>

</configuration>

4）MapReduce配置文件

配置mapred-site.xml

[xiao@hadoop102 hadoop]$ vim mapred-site.xml

文件内容如下：

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

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

<configuration>

<!-- 指定MapReduce程序运行在Yarn上 -->

<property>

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

<value>yarn</value>

</property>

</configuration>

5）配置workers

[xiao@hadoop102 hadoop]$ vim /opt/module/hadoop-3.1.3/etc/hadoop/workers

在该文件中增加如下内容：

hadoop102

hadoop103

hadoop104

注意：该文件中添加的内容结尾不允许有空格，文件中不允许有空行。

## 1.3 配置历史服务器

为了查看程序的历史运行情况，需要配置一下历史服务器。具体配置步骤如下：

**1）配置mapred-site.xml**

[xiao@hadoop102 hadoop]$ vim mapred-site.xml

在该文件里面增加如下配置。

<!-- 历史服务器端地址 -->

<property>

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

<value>hadoop102:10020</value>

</property>

<!-- 历史服务器web端地址 -->

<property>

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

<value>hadoop102:19888</value>

</property>

## 1.4 配置日志的聚集

日志聚集概念：应用运行完成以后，将程序运行日志信息上传到HDFS系统上。

日志聚集功能好处：可以方便的查看到程序运行详情，方便开发调试。

注意：开启日志聚集功能，需要重新启动NodeManager 、ResourceManager和HistoryManager。

开启日志聚集功能具体步骤如下：

**1）配置yarn-site.xml**

[xiao@hadoop102 hadoop]$ vim yarn-site.xml

在该文件里面增加如下配置。

<!-- 开启日志聚集功能 -->

<property>

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

<value>true</value>

</property>

<!-- 设置日志聚集服务器地址 -->

<property>

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

<value>http://hadoop102:19888/jobhistory/logs</value>

</property>

<!-- 设置日志保留时间为7天 -->

<property>

<name>yarn.log-aggregation.retain-seconds</name>

<value>604800</value>

</property>

## 1.5 分发Hadoop

[xiao@hadoop102 hadoop]$ xsync /opt/module/hadoop-3.1.3/

## 1.6 群起集群

**1）启动集群**

（1）**如果集群是第一次启动**，需要在hadoop102节点格式化NameNode（注意格式化之前，一定要先停止上次启动的所有namenode和datanode进程，然后再删除data和log数据）

[xiao@hadoop102 hadoop-3.1.3]$ bin/hdfs namenode -format

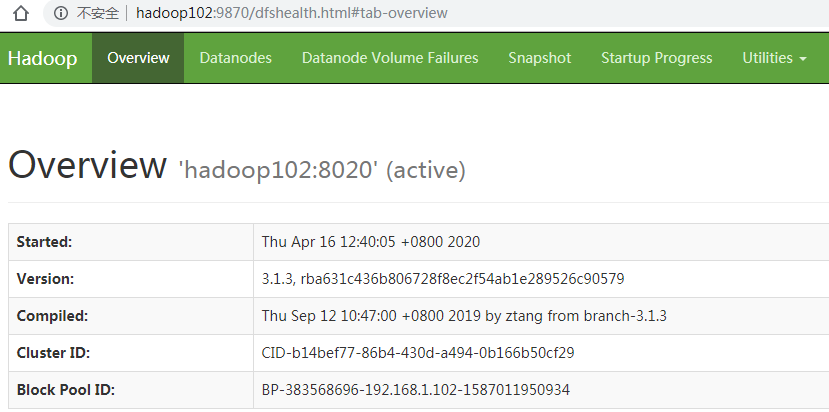
（2）启动HDFS

[xiao@hadoop102 hadoop-3.1.3]$ sbin/start-dfs.sh

（3）**在配置了ResourceManager的节点（hadoop103）**启动YARN

[xiao@hadoop103 hadoop-3.1.3]$ sbin/start-yarn.sh

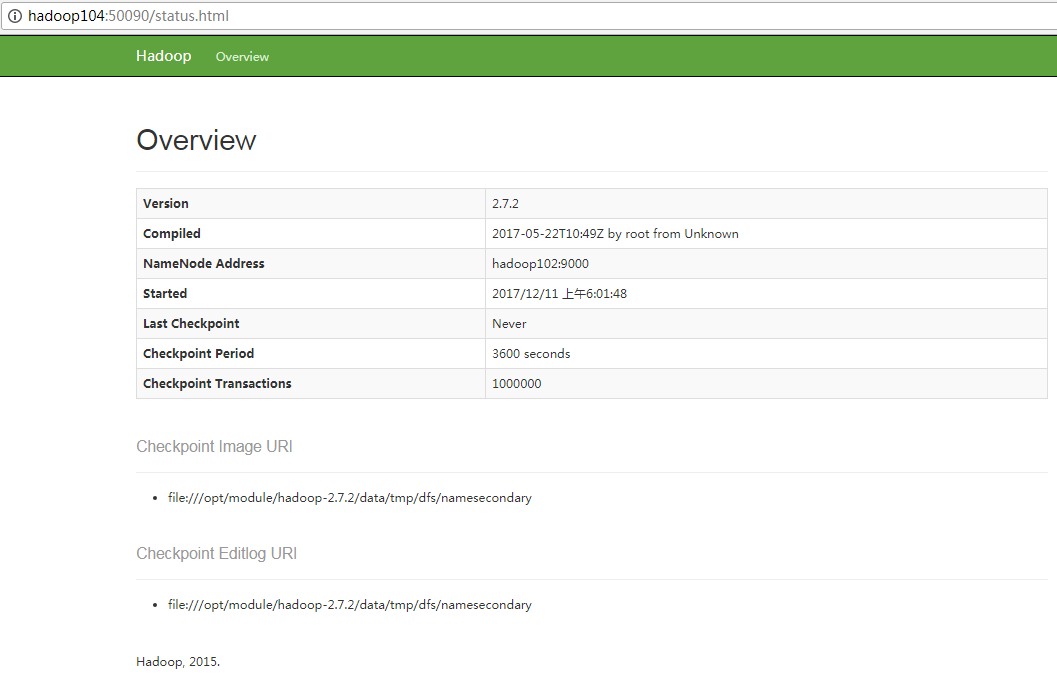
（4）Web端查看HDFS的Web页面：<http://hadoop102:9870/>



（5）Web端查看SecondaryNameNode

（a）浏览器中输入：<http://hadoop104:9868/status.html>

（b）查看SecondaryNameNode信息



## 1.9 Hadoop群起脚本

[xiao@hadoop102 bin]$ pwd

/home/xiao/bin

[xiao@hadoop102 bin]$ vim hdp.sh

输入如下内容：

#!/bin/bash

if [ $# -lt 1 ]

then

echo "No Args Input..."

exit ;

fi

case $1 in

"start")

echo " =================== 启动 hadoop集群 ==================="

echo " --------------- 启动 hdfs ---------------"

ssh hadoop102 "/opt/module/hadoop-3.1.3/sbin/start-dfs.sh"

echo " --------------- 启动 yarn ---------------"

ssh hadoop103 "/opt/module/hadoop-3.1.3/sbin/start-yarn.sh"

echo " --------------- 启动 historyserver ---------------"

ssh hadoop102 "/opt/module/hadoop-3.1.3/bin/mapred --daemon start historyserver"

;;

"stop")

echo " =================== 关闭 hadoop集群 ==================="

echo " --------------- 关闭 historyserver ---------------"

ssh hadoop102 "/opt/module/hadoop-3.1.3/bin/mapred --daemon stop historyserver"

echo " --------------- 关闭 yarn ---------------"

ssh hadoop103 "/opt/module/hadoop-3.1.3/sbin/stop-yarn.sh"

echo " --------------- 关闭 hdfs ---------------"

ssh hadoop102 "/opt/module/hadoop-3.1.3/sbin/stop-dfs.sh"

;;

\*)

echo "Input Args Error..."

;;

esac

[xiao@hadoop102 bin]$ chmod 777 hdp.sh

## 1.10 集群时间同步

时间同步的方式：找一个机器，作为时间服务器，所有的机器与这台集群时间进行定时的同步，比如，每隔十分钟，同步一次时间。

**1）时间服务器配置（必须root用户）**

（0）查看所有节点ntpd服务状态和开机自启动状态

[xiao@hadoop102 ~]$ sudo systemctl status ntpd

[xiao@hadoop102 ~]$ sudo systemctl is-enabled ntpd

（1）在所有节点关闭ntp服务和自启动

[xiao@hadoop102 ~]$ sudo systemctl stop ntpd

[xiao@hadoop102 ~]$ sudo systemctl disable ntpd

（2）修改hadoop102的ntp.conf配置文件

[xiao@hadoop102 ~]$ sudo vim /etc/ntp.conf

修改内容如下

a）修改1（授权192.168.1.0-192.168.1.255网段上的所有机器可以从这台机器上查询和同步时间）

#restrict 192.168.1.0 mask 255.255.255.0 nomodify notrap

为restrict 192.168.1.0 mask 255.255.255.0 nomodify notrap

b）修改2（集群在局域网中，不使用其他互联网上的时间）

server 0.centos.pool.ntp.org iburst

server 1.centos.pool.ntp.org iburst

server 2.centos.pool.ntp.org iburst

server 3.centos.pool.ntp.org iburst

为

**#**server 0.centos.pool.ntp.org iburst

**#**server 1.centos.pool.ntp.org iburst

**#**server 2.centos.pool.ntp.org iburst

**#**server 3.centos.pool.ntp.org iburst

c）添加3（当该节点丢失网络连接，依然可以采用本地时间作为时间服务器为集群中的其他节点提供时间同步）

server 127.127.1.0

fudge 127.127.1.0 stratum 10

（3）修改hadoop102的/etc/sysconfig/ntpd 文件

[xiao@hadoop102 ~]$ sudo vim /etc/sysconfig/ntpd

增加内容如下（让硬件时间与系统时间一起同步）

SYNC\_HWCLOCK=yes

（4）重新启动ntpd服务

[xiao@hadoop102 ~]$ sudo systemctl start ntpd

（5）设置ntpd服务开机启动

[xiao@hadoop102 ~]$ sudo systemctl enable ntpd

**2）其他机器配置（必须root用户）**

（1）在其他机器配置10分钟与时间服务器同步一次

[xiao@hadoop103 ~]$ sudo crontab -e

编写定时任务如下：

\*/10 \* \* \* \* /usr/sbin/ntpdate hadoop102

（2）修改任意机器时间

[xiao@hadoop103 ~]$ sudo date -s "2017-9-11 11:11:11"

（3）十分钟后查看机器是否与时间服务器同步

[xiao@hadoop103 ~]$ sudo date

说明：测试的时候可以将10分钟调整为1分钟，节省时间。

# 第5章：Zookeeper安装

## 1 分布式安装部署

**1）集群规划**

在hadoop102、hadoop103和hadoop104三个节点上部署Zookeeper。

**2）解压安装**

（1）解压Zookeeper安装包到/opt/module/目录下

[xiao@hadoop102 software]$ tar -zxvf zookeeper-3.5.7.tar.gz -C /opt/module/

（2）修改/opt/module/apache-zookeeper-3.5.7-bin名称为zookeeper-3.5.7

[xiao@hadoop102 module]$ mv apache-zookeeper-3.5.7-bin/ zookeeper-3.5.7

（3）同步/opt/module/zookeeper-3.5.7目录内容到hadoop103、hadoop104

[xiao@hadoop102 module]$ xsync zookeeper-3.5.7/

**3）配置服务器编号**

（1）在/opt/module/zookeeper-3.5.7/这个目录下创建zkData

[xiao@hadoop102 zookeeper-3.5.7]$ mkdir zkData

（2）在/opt/module/zookeeper-3.5.7/zkData目录下创建一个myid的文件

[xiao@hadoop102 zkData]$ vi myid

添加myid文件，注意一定要在linux里面创建，在notepad++里面很可能乱码

在文件中添加与server对应的编号：

2

（3）拷贝配置好的zookeeper到其他机器上

[xiao@hadoop102 zkData]$ xsync myid

并分别在hadoop103、hadoop104上修改myid文件中内容为3、4

**4）配置zoo.cfg文件**

（1）重命名/opt/module/zookeeper-3.5.7/conf这个目录下的zoo\_sample.cfg为zoo.cfg

[xiao@hadoop102 conf]$ mv zoo\_sample.cfg zoo.cfg

（2）打开zoo.cfg文件

[xiao@hadoop102 conf]$ vim zoo.cfg

修改数据存储路径配置

dataDir=/opt/module/zookeeper-3.5.7/zkData

增加如下配置

#######################cluster##########################

server.2=hadoop102:2888:3888

server.3=hadoop103:2888:3888

server.4=hadoop104:2888:3888

（3）同步zoo.cfg配置文件

[xiao@hadoop102 conf]$ xsync zoo.cfg

（4）配置参数解读

server.A=B:C:D。

**A**是一个数字，表示这个是第几号服务器；

集群模式下配置一个文件myid，这个文件在dataDir目录下，这个文件里面有一个数据就是A的值，Zookeeper启动时读取此文件，拿到里面的数据与zoo.cfg里面的配置信息比较从而判断到底是哪个server。

**B**是这个服务器的地址；

**C**是这个服务器Follower与集群中的Leader服务器交换信息的端口；

**D**是万一集群中的Leader服务器挂了，需要一个端口来重新进行选举，选出一个新的Leader，而这个端口就是用来执行选举时服务器相互通信的端口。

**5）集群操作**

（1）分别启动Zookeeper

[xiao@hadoop102 zookeeper-3.5.7]$ bin/zkServer.sh start

[xiao@hadoop103 zookeeper-3.5.7]$ bin/zkServer.sh start

[xiao@hadoop104 zookeeper-3.5.7]$ bin/zkServer.sh start

（2）查看状态

[xiao@hadoop102 zookeeper-3.5.7]# bin/zkServer.sh status

JMX enabled by default

Using config: /opt/module/zookeeper-3.5.7/bin/../conf/zoo.cfg

Mode: follower

[xiao@hadoop103 zookeeper-3.5.7]# bin/zkServer.sh status

JMX enabled by default

Using config: /opt/module/zookeeper-3.5.7/bin/../conf/zoo.cfg

Mode: leader

[xiao@hadoop104 zookeeper-3.4.5]# bin/zkServer.sh status

JMX enabled by default

Using config: /opt/module/zookeeper-3.5.7/bin/../conf/zoo.cfg

Mode: follower

## 2 客户端命令行操作

|  |  |
| --- | --- |
| 命令基本语法 | 功能描述 |
| help | 显示所有操作命令 |
| ls path | 使用 ls 命令来查看当前znode的子节点  -w 监听子节点变化  -s 附加次级信息 |
| create | 普通创建  -s 含有序列  -e 临时（重启或者超时消失） |
| get path | 获得节点的值  -w 监听节点内容变化  -s 附加次级信息 |
| set | 设置节点的具体值 |
| stat | 查看节点状态 |
| delete | 删除节点 |
| deleteall | 递归删除节点 |

**启动客户端**

[xiao@hadoop103 zookeeper-3.5.7]$ bin/zkCli.sh

## 3 zookeeper集群启动停止脚本

1）在hadoop102的/home/xiao/bin目录下创建脚本

[xiao@hadoop102 bin]$ vim zk.sh

在脚本中编写如下内容

#!/bin/bash

case $1 in

"start"){

for i in hadoop102 hadoop103 hadoop104

do

echo ---------- zookeeper $i 启动 ------------

ssh $i "/opt/module/zookeeper-3.5.7/bin/zkServer.sh start"

done

};;

"stop"){

for i in hadoop102 hadoop103 hadoop104

do

echo ---------- zookeeper $i 停止 ------------

ssh $i "/opt/module/zookeeper-3.5.7/bin/zkServer.sh stop"

done

};;

"status"){

for i in hadoop102 hadoop103 hadoop104

do

echo ---------- zookeeper $i 状态 ------------

ssh $i "/opt/module/zookeeper-3.5.7/bin/zkServer.sh status"

done

};;

esac

2）增加脚本执行权限

[xiao@hadoop102 bin]$ chmod x zk.sh

3）Zookeeper集群启动脚本

[xiao@hadoop102 module]$ zk.sh start

4）Zookeeper集群停止脚本

[xiao@hadoop102 module]$ zk.sh stop

# 第6章：Kafka安装部署

## 6.1 安装部署

### 6.1.1 集群规划

|  |  |  |
| --- | --- | --- |
| hadoop102 | hadoop103 | hadoop104 |
| zk | zk | zk |
| kafka | kafka | kafka |

### 6.1.2 jar包下载

<http://kafka.apache.org/downloads>

### 6.1.3 集群部署

**1）解压安装包**

[xiao@hadoop102 software]$ tar -zxvf kafka\_2.11-2.4.1.tgz -C /opt/module/

**2）修改解压后的文件名称**

[xiao@hadoop102 module]$ mv kafka\_2.11-2.4.1/ kafka

**3）在/opt/module/kafka目录下创建logs文件夹**

[xiao@hadoop102 kafka]$ mkdir logs

**4）修改配置文件**

[xiao@hadoop102 kafka]$ cd config/

[xiao@hadoop102 config]$ vim server.properties

修改或者增加以下内容：

#broker的全局唯一编号，不能重复（修改）

broker.id=0

#删除topic功能使能（增加）

delete.topic.enable=true

#kafka运行日志存放的路径（修改）

log.dirs=/opt/module/kafka/data

#配置连接Zookeeper集群地址（修改）

zookeeper.connect=hadoop102:2181,hadoop103:2181,hadoop104:2181/kafka

**5）配置环境变量**

[xiao@hadoop102 module]$ sudo vi /etc/profile.d/my\_env.sh

#KAFKA\_HOME

export KAFKA\_HOME=/opt/module/kafka

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

[xiao@hadoop102 module]$ source /etc/profile.d/my\_env.sh

**6）分发安装包**

[xiao@hadoop102 module]$ xsync kafka/

注意：分发之后记得配置其他机器的环境变量

**7）分别在hadoop103和hadoop104上修改配置文件/opt/module/kafka/config/server.properties中的broker.id=1、broker.id=2**

注：broker.id不得重复

**8）启动集群**

依次在hadoop102、hadoop103、hadoop104节点上启动kafka

[xiao@hadoop102 kafka]$ bin/kafka-server-start.sh -daemon /opt/module/kafka/config/server.properties

[xiao@hadoop103 kafka]$ bin/kafka-server-start.sh -daemon /opt/module/kafka/config/server.properties

[xiao@hadoop104 kafka]$ bin/kafka-server-start.sh -daemon /opt/module/kafka/config/server.properties

**9）关闭集群**

[xiao@hadoop102 kafka]$ bin/kafka-server-stop.sh

[xiao@hadoop103 kafka]$ bin/kafka-server-stop.sh

[xiao@hadoop104 kafka]$ bin/kafka-server-stop.sh

**10）kafka群起脚本**

（1）在/home/xiao/bin目录下创建脚本kf.sh

[xiao@hadoop102 bin]$ vim kf.sh

在脚本中填写如下内容

#!/bin/bash

case $1 in

"start"){

for i in hadoop102 hadoop103 hadoop104

do

echo " --------启动 $i Kafka-------"

ssh $i "/opt/module/kafka/bin/kafka-server-start.sh -daemon /opt/module/kafka/config/server.properties "

done

};;

"stop"){

for i in hadoop102 hadoop103 hadoop104

do

echo " --------停止 $i Kafka-------"

ssh $i "/opt/module/kafka/bin/kafka-server-stop.sh"

done

};;

esac

（2）增加脚本执行权限

[xiao@hadoop102 bin]$ chmod 777 kf.sh

（3）kf集群启动脚本

[xiao@hadoop102 module]$ kf.sh start

（4）kf集群停止脚本

[xiao@hadoop102 module]$ kf.sh stop

## 6.2 Kafka命令行操作

**1）查看当前服务器中的所有topic**

[xiao@hadoop102 kafka]$ bin/kafka-topics.sh --zookeeper hadoop102:2181/kafka --list

**2）创建topic**

[xiao@hadoop102 kafka]$ bin/kafka-topics.sh --zookeeper hadoop102:2181/kafka \

--create --replication-factor 3 --partitions 1 --topic first

选项说明：

--topic 定义topic名

--replication-factor 定义副本数

--partitions 定义分区数

**3）删除topic**

[xiao@hadoop102 kafka]$ bin/kafka-topics.sh --zookeeper hadoop102:2181/kafka \

--delete --topic first

需要server.properties中设置delete.topic.enable=true否则只是标记删除。

**4）发送消息**

[xiao@hadoop102 kafka]$ bin/kafka-console-producer.sh \

--broker-list hadoop102:9092 --topic first

>hello world

>xiao xiao

**5）消费消息**

[xiao@hadoop103 kafka]$ bin/kafka-console-consumer.sh \

--bootstrap-server hadoop102:9092 --from-beginning --topic first

[xiao@hadoop103 kafka]$ bin/kafka-console-consumer.sh \

--bootstrap-server hadoop102:9092 --from-beginning --topic first

--from-beginning：会把主题中以往所有的数据都读取出来。

**6）查看某个Topic的详情**

[xiao@hadoop102 kafka]$ bin/kafka-topics.sh --zookeeper hadoop102:2181/kafka \

--describe --topic first

**7）修改分区数**

[xiao@hadoop102 kafka]$bin/kafka-topics.sh --zookeeper hadoop102:2181/kafka --alter --topic first --partitions 6

## 6.3 Kafka集群启停脚本

**1）在/bin目录下创建脚本**kafka**.sh**

[xiao@hadoop102 bin]$ vim kafka.sh

**在脚本中填写如下内容**

#! /bin/bash

case $1 in

"start"){

for i in hadoop102 hadoop103 hadoop104

do

echo " --------启动 $i Kafka-------"

ssh $i "/opt/module/kafka/bin/kafka-server-start.sh -daemon /opt/module/kafka/config/server.properties"

done

};;

"stop"){

for i in hadoop102 hadoop103 hadoop104

do

echo " --------停止 $i Kafka-------"

ssh $i "/opt/module/kafka/bin/kafka-server-stop.sh"

done

};;

esac

**2）增加脚本执行权限**

[xiao@hadoop102 bin]$ chmod +x kafka.sh

**3）**kafka**集群启动脚本**

[xiao@hadoop102 module]$ kafka.sh start

**4）**kafka**集群停止脚本**

[xiao@hadoop102 module]$ kafka.sh stop

# 第7章：Kafka监控（Kafka Eagle）

1）修改kafka启动命令

修改kafka-server-start.sh命令中

if [ "x$KAFKA\_HEAP\_OPTS" = "x" ]; then

export KAFKA\_HEAP\_OPTS="-Xmx1G -Xms1G"

fi

为

if [ "x$KAFKA\_HEAP\_OPTS" = "x" ]; then

export KAFKA\_HEAP\_OPTS="-server -Xms2G -Xmx2G -XX:PermSize=128m -XX:+UseG1GC -XX:MaxGCPauseMillis=200 -XX:ParallelGCThreads=8 -XX:ConcGCThreads=5 -XX:InitiatingHeapOccupancyPercent=70"

export JMX\_PORT="9999"

#export KAFKA\_HEAP\_OPTS="-Xmx1G -Xms1G"

fi

注意：修改之后在启动Kafka之前要分发之其他节点

2）上传压缩包kafka-eagle-bin-1.3.7.tar.gz到集群/opt/software目录

3）解压到本地

[xiao@hadoop102 software]$ tar -zxvf kafka-eagle-bin-1.3.7.tar.gz

4）进入刚才解压的目录

[xiao@hadoop102 kafka-eagle-bin-1.3.7]$ ll

总用量 82932

-rw-rw-r--. 1 xiao xiao 84920710 8月 13 23:00 kafka-eagle-web-1.3.7-bin.tar.gz

5）将kafka-eagle-web-1.3.7-bin.tar.gz解压至/opt/module

[xiao@hadoop102 kafka-eagle-bin-1.3.7]$ tar -zxvf kafka-eagle-web-1.3.7-bin.tar.gz -C /opt/module/

6）修改名称

[xiao@hadoop102 module]$ mv kafka-eagle-web-1.3.7/ eagle

7）给启动文件执行权限

[xiao@hadoop102 eagle]$ cd bin/

[xiao@hadoop102 bin]$ ll

总用量 12

-rw-r--r--. 1 xiao xiao 1848 8月 22 2017 ke.bat

-rw-r--r--. 1 xiao xiao 7190 7月 30 20:12 ke.sh

[xiao@hadoop102 bin]$ chmod 777 ke.sh

8）修改配置文件

######################################

# multi zookeeper&kafka cluster list

######################################

kafka.eagle.zk.cluster.alias=cluster1

cluster1.zk.list=hadoop102:2181,hadoop103:2181,hadoop104:2181/kafka

######################################

# kafka offset storage

######################################

cluster1.kafka.eagle.offset.storage=kafka

######################################

# enable kafka metrics

######################################

kafka.eagle.metrics.charts=true

kafka.eagle.sql.fix.error=false

######################################

# kafka jdbc driver address

######################################

kafka.eagle.driver=com.mysql.jdbc.Driver

kafka.eagle.url=jdbc:mysql://hadoop102:3306/ke?useUnicode=true&characterEncoding=UTF-8&zeroDateTimeBehavior=convertToNull

kafka.eagle.username=root

kafka.eagle.password=000000

9）添加环境变量

export KE\_HOME=/opt/module/eagle

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

注意：source /etc/profile

10）启动

[xiao@hadoop102 eagle]$ bin/ke.sh start

... ...

... ...

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Kafka Eagle Service has started success.

\* Welcome, Now you can visit 'http://192.168.9.102:8048/ke'

\* Account:admin ,Password:123456

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* <Usage> ke.sh [start|status|stop|restart|stats] </Usage>

\* <Usage> https://www.kafka-eagle.org/ </Usage>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[xiao@hadoop102 eagle]$

注意：启动之前需要先启动ZK以及KAFKA

11）登录页面查看监控数据

<http://192.168.9.102:8048/ke>

# 第8章：Flume安装

### 8.1 安装地址

（1） Flume官网地址：<http://flume.apache.org/>

（2）文档查看地址：<http://flume.apache.org/FlumeUserGuide.html>

（3）下载地址：http://archive.apache.org/dist/flume/

### 8.2 安装部署

（1）将apache-flume-1.9.0-bin.tar.gz上传到linux的/opt/software目录下

（2）解压apache-flume-1.9.0-bin.tar.gz到/opt/module/目录下

[xiao@hadoop102 software]$ tar -zxf /opt/software/apache-flume-1.9.0-bin.tar.gz -C /opt/module/

（3）修改apache-flume-1.9.0-bin的名称为flume

[xiao@hadoop102 module]$ mv /opt/module/apache-flume-1.9.0-bin /opt/module/flume

（4）将lib文件夹下的guava-11.0.2.jar删除以兼容Hadoop 3.1.3

[xiao@hadoop102 module]$ rm /opt/module/flume/lib/guava-11.0.2.jar

注意：删除guava-11.0.2.jar的服务器节点，一定要配置hadoop环境变量。否则会报如下异常。

Caused by: java.lang.ClassNotFoundException: com.google.common.collect.Lists

at java.net.URLClassLoader.findClass(URLClassLoader.java:382)

at java.lang.ClassLoader.loadClass(ClassLoader.java:424)

at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:349)

at java.lang.ClassLoader.loadClass(ClassLoader.java:357)

... 1 more

（5）将flume/conf下的flume-env.sh.template文件修改为flume-env.sh，并配置flume-env.sh文件

[xiao@hadoop102 conf]$ mv flume-env.sh.template flume-env.sh

[xiao@hadoop102 conf]$ vi flume-env.sh

export JAVA\_HOME=/opt/module/jdk1.8.0\_212

### 8.3 消费日志flume

#### 8.3.1 项目经验之Flume组件选型

1）Source

（1）Taildir Source相比Exec Source、Spooling Directory Source的优势

TailDir Source：断点续传、多目录。Flume1.6以前需要自定义Source记录每次读取文件位置，实现断点续传。 不会丢数据，但是有可能会导致数据重复。

Exec Source可以实时搜集数据，但是在Flume不运行或者Shell命令出错的情况下，数据将会丢失。

Spooling Directory Source监控目录，支持断点续传。

（2）batchSize大小如何设置？

答：Event 1K左右时，500-1000合适（默认为100）

2）Channel

采用Kafka Channel，省去了Sink，提高了效率。KafkaChannel数据存储在Kafka里面，所以数据是存储在磁盘中。

注意在Flume1.7以前，Kafka Channel很少有人使用，因为发现parseAsFlumeEvent这个配置起不了作用。也就是无论parseAsFlumeEvent配置为true还是false，都会转为Flume Event。这样的话，造成的结果是，会始终都把Flume的headers中的信息混合着内容一起写入Kafka的消息中，这显然不是我所需要的，我只是需要把内容写入即可。

#### 8.3.2 日志采集Flume配置

1）Flume配置分析

Flume直接读log日志的数据，log日志的格式是app.yyyy-mm-dd.log。

2）Flume的具体配置如下：

（1）在/opt/module/flume/conf目录下创建file-flume-kafka.conf文件

[xiao@hadoop102 conf]$ vim file-flume-kafka.conf

在文件配置如下内容

#为各组件命名

a1.sources = r1

a1.channels = c1

#描述source

a1.sources.r1.type = TAILDIR

a1.sources.r1.filegroups = f1

a1.sources.r1.filegroups.f1 = /opt/module/applog/log/app.\*

a1.sources.r1.positionFile = /opt/module/flume/taildir\_position.json

a1.sources.r1.interceptors = i1

a1.sources.r1.interceptors.i1.type = com.xiao.flume.interceptor.ETLInterceptor$Builder

#描述channel

a1.channels.c1.type = org.apache.flume.channel.kafka.KafkaChannel

a1.channels.c1.kafka.bootstrap.servers = hadoop102:9092,hadoop103:9092

a1.channels.c1.kafka.topic = topic\_log

a1.channels.c1.parseAsFlumeEvent = false

#绑定source和channel以及sink和channel的关系

a1.sources.r1.channels = c1

注意：com.xiao.flume.interceptor.ETLInterceptor是自定义的拦截器的全类名。需要根据用户自定义的拦截器做相应修改。

#### 8.3.3 Flume拦截器

1）创建Maven工程flume-interceptor

2）创建包名：com.xiao.flume.interceptor

3）在pom.xml文件中添加如下配置

<dependencies>  
 <dependency>  
 <groupId>org.apache.flume</groupId>  
 <artifactId>flume-ng-core</artifactId>  
 <version>1.9.0</version>  
 <scope>provided</scope>  
 </dependency>  
  
 <dependency>  
 <groupId>com.alibaba</groupId>  
 <artifactId>fastjson</artifactId>  
 <version>1.2.62</version>  
 </dependency>  
</dependencies>  
  
<build>  
 <plugins>  
 <plugin>  
 <artifactId>maven-compiler-plugin</artifactId>  
 <version>2.3.2</version>  
 <configuration>  
 <source>1.8</source>  
 <target>1.8</target>  
 </configuration>  
 </plugin>  
 <plugin>  
 <artifactId>maven-assembly-plugin</artifactId>  
 <configuration>  
 <descriptorRefs>  
 <descriptorRef>jar-with-dependencies</descriptorRef>  
 </descriptorRefs>  
 </configuration>  
 <executions>  
 <execution>  
 <id>make-assembly</id>  
 <phase>package</phase>  
 <goals>  
 <goal>single</goal>  
 </goals>  
 </execution>  
 </executions>  
 </plugin>  
 </plugins>  
</build>

注意：scope中provided的含义是编译时用该jar包。打包时不用。因为集群上已经存在flume的jar包。只是本地编译时用一下。

4）在com.xiao.flume.interceptor包下创建JSONUtils类

package com.xiao.flume.interceptor;  
  
import com.alibaba.fastjson.JSON;  
import com.alibaba.fastjson.JSONException;  
  
public class JSONUtils {  
 public static boolean isJSONValidate(String log){  
 try {  
 JSON.parse(log);  
 return true;  
 }catch (JSONException e){  
 return false;  
 }  
 }  
}

5）在com.xiao.flume.interceptor包下创建ETLInterceptor类

package com.xiao.flume.interceptor;

import com.alibaba.fastjson.JSON;

import org.apache.flume.Context;

import org.apache.flume.Event;

import org.apache.flume.interceptor.Interceptor;

import java.nio.charset.StandardCharsets;

import java.util.Iterator;

import java.util.List;

public class ETLInterceptor implements Interceptor {

@Override

public void initialize() {

}

@Override

public Event intercept(Event event) {

byte[] body = event.getBody();

String log = new String(body, StandardCharsets.UTF\_8);

if (JSONUtils.isJSONValidate(log)) {

return event;

} else {

return null;

}

}

@Override

public List<Event> intercept(List<Event> list) {

Iterator<Event> iterator = list.iterator();

while (iterator.hasNext()){

Event next = iterator.next();

if(intercept(next)==null){

iterator.remove();

}

}

return list;

}

public static class Builder implements Interceptor.Builder{

@Override

public Interceptor build() {

return new ETLInterceptor();

}

@Override

public void configure(Context context) {

}

}

@Override

public void close() {

}

}

6）打包



7）需要先将打好的包放入到hadoop102的/opt/module/flume/lib文件夹下面。

[xiao@hadoop102 lib]$ ls | grep interceptor

flume-interceptor-1.0-SNAPSHOT-jar-with-dependencies.jar

8）分发Flume到hadoop103、hadoop104

[xiao@hadoop102 module]$ xsync flume/

9）分别在hadoop102、hadoop103上启动Flume

[xiao@hadoop102 flume]$ bin/flume-ng agent --name a1 --conf-file conf/file-flume-kafka.conf &

[xiao@hadoop103 flume]$ bin/flume-ng agent --name a1 --conf-file conf/file-flume-kafka.conf &

#### 8.3.4 日志采集Flume启动停止脚本

1）在/home/xiao/bin目录下创建脚本f1.sh

[xiao@hadoop102 bin]$ vim f1.sh

在脚本中填写如下内容

#! /bin/bash

case $1 in

"start"){

for i in hadoop102 hadoop103

do

echo " --------启动 $i 采集flume-------"

ssh $i "nohup /opt/module/flume/bin/flume-ng agent --conf-file /opt/module/flume/conf/file-flume-kafka.conf --name a1 -Dflume.root.logger=INFO,LOGFILE >/opt/module/flume/log1.txt 2>&1 &"

done

};;

"stop"){

for i in hadoop102 hadoop103

do

echo " --------停止 $i 采集flume-------"

ssh $i "ps -ef | grep file-flume-kafka | grep -v grep |awk '{print \$2}' | xargs -n1 kill -9 "

done

};;

esac

说明1：nohup，该命令可以在你退出帐户/关闭终端之后继续运行相应的进程。nohup就是不挂起的意思，不挂断地运行命令。

说明2：awk默认分隔符为空格

说明3：xargs表示取出前面命令运行的结果，作为后面命令的输入参数。

2）增加脚本执行权限

[xiao@hadoop102 bin]$ chmod +x f1.sh

3）f1集群启动脚本

[xiao@hadoop102 module]$ f1.sh start

4）f1集群停止脚本

[xiao@hadoop102 module]$ f1.sh stop

### 8.4 消费kafka主题flume

#### 8.4.1 项目经验之Flume组件选型

1）FileChannel和MemoryChannel区别

MemoryChannel传输数据速度更快，但因为数据保存在JVM的堆内存中，Agent进程挂掉会导致数据丢失，适用于对数据质量要求不高的需求。

FileChannel传输速度相对于Memory慢，但数据安全保障高，Agent进程挂掉也可以从失败中恢复数据。

选型：

金融类公司、对钱要求非常准确的公司通常会选择FileChannel

传输的是普通日志信息（京东内部一天丢100万-200万条，这是非常正常的），通常选择MemoryChannel。

2）FileChannel优化

通过配置dataDirs指向多个路径，每个路径对应不同的硬盘，增大Flume吞吐量。

官方说明如下：

Comma separated list of directories for storing log files. Using multiple directories on separate disks can improve file channel peformance

checkpointDir和backupCheckpointDir也尽量配置在不同硬盘对应的目录中，保证checkpoint坏掉后，可以快速使用backupCheckpointDir恢复数据。

3）Sink：HDFS Sink

（1）HDFS存入大量小文件，有什么影响？

**元数据层面：**每个小文件都有一份元数据，其中包括文件路径，文件名，所有者，所属组，权限，创建时间等，这些信息都保存在Namenode内存中。所以小文件过多，会占用Namenode服务器大量内存，影响Namenode性能和使用寿命。

**计算层面：**默认情况下MR会对每个小文件启用一个Map任务计算，非常影响计算性能。同时也影响磁盘寻址时间。

（2）HDFS小文件处理

官方默认的这三个参数配置写入HDFS后会产生小文件，hdfs.rollInterval、hdfs.rollSize、hdfs.rollCount。

基于以上hdfs.rollInterval=3600，hdfs.rollSize=134217728，hdfs.rollCount =0几个参数综合作用，效果如下：

（1）文件在达到128M时会滚动生成新文件

（2）文件创建超3600秒时会滚动生成新文件

#### 8.4.2 日志消费Flume配置

1）Flume配置分析

2）Flume的具体配置如下：

（1）在hadoop104的/opt/module/flume/conf目录下创建kafka-flume-hdfs.conf文件

[xiao@hadoop104 conf]$ vim kafka-flume-hdfs.conf

在文件配置如下内容

## 组件

a1.sources=r1

a1.channels=c1

a1.sinks=k1

## source1

a1.sources.r1.type = org.apache.flume.source.kafka.KafkaSource

a1.sources.r1.batchSize = 5000

a1.sources.r1.batchDurationMillis = 2000

a1.sources.r1.kafka.bootstrap.servers = hadoop102:9092,hadoop103:9092,hadoop104:9092

a1.sources.r1.kafka.topics=topic\_log

a1.sources.r1.interceptors = i1

a1.sources.r1.interceptors.i1.type = com.xiao.flume.interceptor.TimeStampInterceptor$Builder

## channel1

a1.channels.c1.type = file

a1.channels.c1.checkpointDir = /opt/module/flume/checkpoint/behavior1

a1.channels.c1.dataDirs = /opt/module/flume/data/behavior1/

a1.channels.c1.maxFileSize = 2146435071

a1.channels.c1.capacity = 1000000

a1.channels.c1.keep-alive = 6

## sink1

a1.sinks.k1.type = hdfs

a1.sinks.k1.hdfs.path = /gmall/log/topic\_log/%Y-%m-%d

a1.sinks.k1.hdfs.filePrefix = log-

a1.sinks.k1.hdfs.round = false

a1.sinks.k1.hdfs.rollInterval = 10

a1.sinks.k1.hdfs.rollSize = 134217728

a1.sinks.k1.hdfs.rollCount = 0

## 控制输出文件是原生文件。

a1.sinks.k1.hdfs.fileType = CompressedStream

a1.sinks.k1.hdfs.codeC = lzop

## 拼装

a1.sources.r1.channels = c1

a1.sinks.k1.channel= c1

#### 8.4.3 Flume拦截器

由于Flume默认会用Linux系统时间，作为输出到HDFS路径的时间。如果数据是23:59分产生的。Flume消费Kafka里面的数据时，有可能已经是第二天了，那么这部分数据会被发往第二天的HDFS路径。我们希望的是根据日志里面的实际时间，发往HDFS的路径，所以下面拦截器作用是获取日志中的实际时间。

解决的思路：拦截json日志，通过fastjson框架解析json，获取实际时间ts。将获取的ts时间写入拦截器header头，header的key必须是timestamp，因为Flume框架会根据这个key的值识别为时间，写入到HDFS。

1）在com.xiao.flume.interceptor包下创建TimeStampInterceptor类

package com.xiao.flume.interceptor;

import com.alibaba.fastjson.JSONObject;

import org.apache.flume.Context;

import org.apache.flume.Event;

import org.apache.flume.interceptor.Interceptor;

import java.nio.charset.StandardCharsets;

import java.util.ArrayList;

import java.util.List;

import java.util.Map;

public class TimeStampInterceptor implements Interceptor {

@Override

public void initialize() {

}

@Override

public Event intercept(Event event) {

//从event的body里获取一条日志

byte[] body = event.getBody();

String log = new String(body);

//将日志转换成json

JSONObject jsonObject = JSON.parseObject(log);

if(jsonObject.containsKey("ts")){

//获取日志里面的ts时间戳,放到event的header里

String ts = jsonObject.getString("ts");

event.getHeaders().put("timestamp",ts);

}

return event;

}

@Override

public List<Event> intercept(List<Event> list) {

for (Event event : list) {

intercept(event);

}

return list;

}

@Override

public void close() {

}

public static class Builder implements Interceptor.Builder {

@Override

public Interceptor build() {

return new TimeStampInterceptor();

}

@Override

public void configure(Context context) {

}

}

}

2）重新打包



3）需要先将打好的包放入到hadoop102的/opt/module/flume/lib文件夹下面。

[xiao@hadoop102 lib]$ ls | grep interceptor

flume-interceptor-1.0-SNAPSHOT-jar-with-dependencies.jar

4）分发Flume到hadoop103、hadoop104

[xiao@hadoop102 module]$ xsync flume/

#### 8.4.4 日志消费Flume启动停止脚本

1）在/home/xiao/bin目录下创建脚本f2.sh

[xiao@hadoop102 bin]$ vim f2.sh

在脚本中填写如下内容

#! /bin/bash

case $1 in

"start"){

for i in hadoop104

do

echo " --------启动 $i 消费flume-------"

ssh $i "nohup /opt/module/flume/bin/flume-ng agent --conf-file /opt/module/flume/conf/kafka-flume-hdfs.conf --name a1 -Dflume.root.logger=INFO,LOGFILE >/opt/module/flume/log2.txt 2>&1 &"

done

};;

"stop"){

for i in hadoop104

do

echo " --------停止 $i 消费flume-------"

ssh $i "ps -ef | grep kafka-flume-hdfs | grep -v grep |awk '{print \$2}' | xargs -n1 kill -9"

done

};;

esac

2）增加脚本执行权限

[xiao@hadoop102 bin]$ chmod +x f2.sh

3）f2集群启动脚本

[xiao@hadoop102 module]$ f2.sh start

4）f2集群停止脚本

[xiao@hadoop102 module]$ f2.sh stop

#### 8.4.5 项目经验之Flume内存优化

1）问题描述：如果启动消费Flume抛出如下异常

ERROR hdfs.HDFSEventSink: process failed

java.lang.OutOfMemoryError: GC overhead limit exceeded

2）解决方案步骤：

（1）在hadoop102服务器的/opt/module/flume/conf/flume-env.sh文件中增加如下配置

export JAVA\_OPTS="-Xms100m -Xmx2000m -Dcom.sun.management.jmxremote"

（2）同步配置到hadoop103、hadoop104服务器

[xiao@hadoop102 conf]$ xsync flume-env.sh

3）Flume内存参数设置及优化

JVM heap一般设置为4G或更高

-Xmx与-Xms最好设置一致，减少内存抖动带来的性能影响，如果设置不一致容易导致频繁fullgc。

-Xms表示JVM Heap（堆内存）最小尺寸，初始分配；-Xmx 表示JVM Heap(堆内存)最大允许的尺寸，按需分配。如果不设置一致，容易在初始化时，由于内存不够，频繁触发fullgc。

# 第9章：MySQL安装

### 9.1 安装包准备

**1）将安装包和JDBC驱动上传到/opt/software，共计6个**

01\_mysql-community-common-5.7.16-1.el7.x86\_64.rpm

02\_mysql-community-libs-5.7.16-1.el7.x86\_64.rpm

03\_mysql-community-libs-compat-5.7.16-1.el7.x86\_64.rpm

04\_mysql-community-client-5.7.16-1.el7.x86\_64.rpm

05\_mysql-community-server-5.7.16-1.el7.x86\_64.rpm

mysql-connector-java-5.1.27-bin.jar

**2）如果是虚拟机按照如下步骤执行**

（1）卸载自带的Mysql-libs（如果之前安装过mysql，要全都卸载掉）

[xiao@hadoop102 software]$ rpm -qa | grep -i -E mysql\|mariadb | xargs -n1 sudo rpm -e --nodeps

**3）如果是阿里云服务器按照如下步骤执行**

（1）卸载MySQL依赖，虽然机器上没有装MySQL，但是这一步不可少

[xiao@hadoop102 software]# sudo yum remove mysql-libs

（2）下载依赖并安装

[xiao@hadoop102 software]# sudo yum install libaio

[xiao@hadoop102 software]# sudo yum -y install autoconf

### 9.2 安装MySQL

**1）安装mysql依赖**

[xiao@hadoop102 software]$ sudo rpm -ivh 01\_mysql-community-common-5.7.16-1.el7.x86\_64.rpm

[xiao@hadoop102 software]$ sudo rpm -ivh 02\_mysql-community-libs-5.7.16-1.el7.x86\_64.rpm

[xiao@hadoop102 software]$ sudo rpm -ivh 03\_mysql-community-libs-compat-5.7.16-1.el7.x86\_64.rpm

**2）安装mysql-client**

[xiao@hadoop102 software]$ sudo rpm -ivh 04\_mysql-community-client-5.7.16-1.el7.x86\_64.rpm

**3）安装mysql-server**

[xiao@hadoop102 software]$ sudo rpm -ivh 05\_mysql-community-server-5.7.16-1.el7.x86\_64.rpm

**注意：如果报如下错误，这是由于yum安装了旧版本的GPG keys所造成，从rpm版本4.1后，在安装或升级软件包时会自动检查软件包的签名。**

warning: 05\_mysql-community-server-5.7.16-1.el7.x86\_64.rpm: Header V3 DSA/SHA1 Signature, key ID 5072e1f5: NOKEY

error: Failed dependencies:

libaio.so.1()(64bit) is needed by mysql-community-server-5.7.16-1.el7.x86\_64

**解决办法**

[xiao@hadoop102 software]$ sudo rpm -ivh 05\_mysql-community-server-5.7.16-1.el7.x86\_64.rpm --force --nodeps

**4）启动mysql**

[xiao@hadoop102 software]$ sudo systemctl start mysqld

**5）查看mysql密码**

[xiao@hadoop102 software]$ sudo cat /var/log/mysqld.log | grep password

### 9.3 配置MySQL

配置只要是root用户+密码，在任何主机上都能登录MySQL数据库。

**1）用刚刚查到的密码进入mysql（如果报错，给密码加单引号）**

[xiao@hadoop102 software]$ mysql -uroot -p'password'

**2）设置复杂密码（由于mysql密码策略，此密码必须足够复杂）**

mysql> set password=password("Qs23=zs32");

**3）更改mysql密码策略**

mysql> set global validate\_password\_length=4;

mysql> set global validate\_password\_policy=0;

**4）设置简单好记的密码**

mysql> set password=password("123456");

**5）进入msyql库**

mysql> use mysql;

**6）查询user表**

mysql> select user, host from user;

**7）修改user表，把Host表内容修改为%**

mysql> update user set host="%" where user="root";

**8）刷新**

mysql> flush privileges;

**9）退出**

mysql> quit;

# 第10章：Sqoop安装

### 10.1 下载并解压

1）sqoop官网地址：<http://sqoop.apache.org/docs/1.4.7/index.html>

2）下载地址：<http://mirrors.hust.edu.cn/apache/sqoop/1.4.7/>

3）上传安装包sqoop-1.4.7.bin\_\_hadoop-2.6.0.tar.gz到hadoop102的/opt/software路径中

4）解压sqoop安装包到指定目录，如：

[xiao@hadoop102 software]$ tar -zxvf sqoop-1.4.7.bin\_\_hadoop-2.6.0.tar.gz -C /opt/module/

5）修改名称为sqoop：

[xiao@hadoop102 module]$ mv sqoop-1.4.7.bin\_\_hadoop-2.6.0/ sqoop

### 10.2 修改配置文件

1）进入到/opt/module/sqoop/conf目录，重命名配置文件

[xiao@hadoop102 conf]$ mv sqoop-env-template.sh sqoop-env.sh

2）修改配置文件

[xiao@hadoop102 conf]$ vim sqoop-env.sh

增加如下内容

export HADOOP\_COMMON\_HOME=/opt/module/hadoop-3.1.3

export HADOOP\_MAPRED\_HOME=/opt/module/hadoop-3.1.3

export HIVE\_HOME=/opt/module/hive

export ZOOKEEPER\_HOME=/opt/module/zookeeper-3.5.7

export ZOOCFGDIR=/opt/module/zookeeper-3.5.7/conf

### 10.3 拷贝JDBC驱动

1）将mysql-connector-java-5.1.48.jar 上传到/opt/software路径

2）进入到/opt/software/路径，拷贝jdbc驱动到sqoop的lib目录下。

[xiao@hadoop102 software]$ cp /opt/software/mysql-connector-java-5.1.48.jar /opt/module/sqoop/lib/

### 10.4 验证Sqoop

我们可以通过某一个command来验证sqoop配置是否正确：

[xiao@hadoop102 sqoop]$ bin/sqoop help

出现一些Warning警告（警告信息已省略），并伴随着帮助命令的输出：

Available commands:

codegen Generate code to interact with database records

create-hive-table Import a table definition into Hive

eval Evaluate a SQL statement and display the results

export Export an HDFS directory to a database table

help List available commands

import Import a table from a database to HDFS

import-all-tables Import tables from a database to HDFS

import-mainframe Import datasets from a mainframe server to HDFS

job Work with saved jobs

list-databases List available databases on a server

list-tables List available tables in a database

merge Merge results of incremental imports

metastore Run a standalone Sqoop metastore

version Display version information

### 10.5 测试Sqoop是否能够成功连接数据库

[xiao@hadoop102 sqoop]$ bin/sqoop list-databases --connect jdbc:mysql://hadoop102:3306/ --username root --password 123456

出现如下输出：

information\_schema

metastore

mysql

oozie

performance\_schema

### 10.6 Sqoop基本使用

将mysql中user\_info表数据导入到hdfs的/test路径

bin/sqoop import \

--connect jdbc:mysql://hadoop102:3306/gmall \

--username root \

--password 123456 \

--table user\_info \

--columns id,login\_name \

--where "id>=10 and id<=30" \

--target-dir /test \

--delete-target-dir \

--fields-terminated-by '\t' \

--num-mappers 2 \

--split-by id

# 第11章：Hive安装

## 11.1 Hive安装部署

**1）把apache-hive-3.1.2-bin.tar.gz上传到linux的/opt/software目录下**

**2）解压apache-hive-3.1.2-bin.tar.gz到/opt/module/目录下面**

[xiao@hadoop102 software]$ tar -zxvf /opt/software/apache-hive-3.1.2-bin.tar.gz -C /opt/module/

**3）修改apache-hive-3.1.2-bin.tar.gz的名称为hive**

[xiao@hadoop102 software]$ mv /opt/module/apache-hive-3.1.2-bin/ /opt/module/hive

**4）修改/etc/profile.d/my\_env.sh，添加环境变量**

[xiao@hadoop102 software]$ sudo vim /etc/profile.d/my\_env.sh

添加内容

#HIVE\_HOME

export HIVE\_HOME=/opt/module/hive

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

重启Xshell对话框或者source一下 /etc/profile.d/my\_env.sh文件，使环境变量生效

[xiao@hadoop102 software]$ source /etc/profile.d/my\_env.sh

**5）解决日志Jar包冲突，进入/opt/module/hive/lib目录**

[xiao@hadoop102 lib]$ mv log4j-slf4j-impl-2.10.0.jar log4j-slf4j-impl-2.10.0.jar.bak

## 11.2 Hive元数据配置到MySQL

### 11.2.1 拷贝驱动

将MySQL的JDBC驱动拷贝到Hive的lib目录下

[xiao@hadoop102 lib]$ cp /opt/software/mysql-connector-java-5.1.48.jar /opt/module/hive/lib/

### 11.2.2 配置Metastore到MySQL

在$HIVE\_HOME/conf目录下新建hive-site.xml文件

[xiao@hadoop102 conf]$ vim hive-site.xml

添加如下内容

<?xml version="1.0"?>

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

<configuration>

<property>

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

<value>jdbc:mysql://hadoop102:3306/metastore?useSSL=false</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>123456</value>

</property>

<property>

<name>hive.metastore.warehouse.dir</name>

<value>/user/hive/warehouse</value>

</property>

<property>

<name>hive.metastore.schema.verification</name>

<value>false</value>

</property>

<property>

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

<value>10000</value>

</property>

<property>

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

<value>hadoop102</value>

</property>

<property>

<name>hive.metastore.event.db.notification.api.auth</name>

<value>false</value>

</property>

<property>

<name>hive.cli.print.header</name>

<value>true</value>

</property>

<property>

<name>hive.cli.print.current.db</name>

<value>true</value>

</property>

</configuration>

## 11.3 启动Hive

### 11.3.1 初始化元数据库

**1）登陆MySQL**

[xiao@hadoop102 conf]$ mysql -uroot –p123456

**2）新建Hive元数据库**

mysql> create database metastore;

mysql> quit;

**3）初始化Hive元数据库**

[xiao@hadoop102 conf]$ schematool -initSchema -dbType mysql -verbose

### 11.3.2 启动hive客户端

**1）启动Hive客户端**

[xiao@hadoop102 hive]$ bin/hive

**2）查看一下数据库**

hive (default)> show databases;

OK

database\_name

default

# 第12章：Hive环境准备

### 12.1 Hive引擎简介

Hive引擎包括：默认MR、Tez、Spark

Hive on Spark：Hive既作为存储元数据又负责SQL的解析优化，语法是HQL语法，执行引擎变成了Spark，Spark负责采用RDD执行。

Spark on Hive : Hive只作为存储元数据，Spark负责SQL解析优化，语法是Spark SQL语法，Spark负责采用RDD执行。

### 12.2 Hive on Spark配置

**1）兼容性说明**

**注意**：官网下载的Hive3.1.2和Spark3.0.0默认是不兼容的。因为Hive3.1.2支持的Spark版本是2.4.5，所以需要我们重新编译Hive3.1.2版本。

编译步骤：官网下载Hive3.1.2源码，修改pom文件中引用的Spark版本为3.0.0，如果编译通过，直接打包获取jar包。如果报错，就根据提示，修改相关方法，直到不报错，打包获取jar包。

**2）在Hive所在节点部署Spark**

如果之前已经部署了Spark，则该步骤可以跳过，但要检查SPARK\_HOME的环境变量配置是否正确。

（1）Spark官网下载jar包地址：

<http://spark.apache.org/downloads.html>

（2）上传并解压解压spark-3.0.0-bin-hadoop3.2.tgz

[xiao@hadoop102 software]$ tar -zxvf spark-3.0.0-bin-hadoop3.2.tgz -C /opt/module/

[xiao@hadoop102 software]$ mv /opt/module/spark-3.0.0-bin-hadoop3.2 /opt/module/spark

（3）配置SPARK\_HOME环境变量

[xiao@hadoop102 software]$ sudo vim /etc/profile.d/my\_env.sh

添加如下内容

# SPARK\_HOME

export SPARK\_HOME=/opt/module/spark

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

source 使其生效

[xiao@hadoop102 software]$ source /etc/profile.d/my\_env.sh

（4）新建spark配置文件

[xiao@hadoop102 software]$ vim /opt/module/hive/conf/spark-defaults.conf

添加如下内容（在执行任务时，会根据如下参数执行）

spark.master yarn

spark.eventLog.enabled true

spark.eventLog.dir hdfs://hadoop102:8020/spark-history

spark.executor.memory 1g

spark.driver.memory 1g

（5）在HDFS创建如下路径，用于存储历史日志

[xiao@hadoop102 software]$ hadoop fs -mkdir /spark-history

**3）向HDFS上传Spark纯净版jar包**

说明1：由于Spark3.0.0非纯净版默认支持的是hive2.3.7版本，直接使用会和安装的Hive3.1.2出现兼容性问题。所以采用Spark纯净版jar包，不包含hadoop和hive相关依赖，避免冲突。

说明2：Hive任务最终由Spark来执行，Spark任务资源分配由Yarn来调度，该任务有可能被分配到集群的任何一个节点。所以需要将Spark的依赖上传到HDFS集群路径，这样集群中任何一个节点都能获取到。

（1）上传并解压spark-3.0.0-bin-without-hadoop.tgz

[xiao@hadoop102 software]$ tar -zxvf /opt/software/spark-3.0.0-bin-without-hadoop.tgz

（2）上传Spark纯净版jar包到HDFS

[xiao@hadoop102 software]$ hadoop fs -mkdir /spark-jars

[xiao@hadoop102 software]$ hadoop fs -put spark-3.0.0-bin-without-hadoop/jars/\* /spark-jars

**4）修改hive-site.xml文件**

[xiao@hadoop102 ~]$ vim /opt/module/hive/conf/hive-site.xml

添加如下内容

<!--Spark依赖位置（注意：端口号8020必须和namenode的端口号一致）-->

<property>

<name>spark.yarn.jars</name>

<value>hdfs://hadoop102:8020/spark-jars/\*</value>

</property>

<!--Hive执行引擎-->

<property>

<name>hive.execution.engine</name>

<value>spark</value>

</property>

<!--Hive和Spark连接超时时间-->

<property>

<name>hive.spark.client.connect.timeout</name>

<value>10000ms</value>

</property>

注意：hive.spark.client.connect.timeout的默认值是1000ms，如果执行hive的insert语句时，抛如下异常，可以调大该参数到10000ms

FAILED: SemanticException Failed to get a spark session: org.apache.hadoop.hive.ql.metadata.HiveException: Failed to create Spark client for Spark session d9e0224c-3d14-4bf4-95bc-ee3ec56df48e

### 12.3 Hive on Spark测试

（1）启动hive客户端

[xiao@hadoop102 hive]$ bin/hive

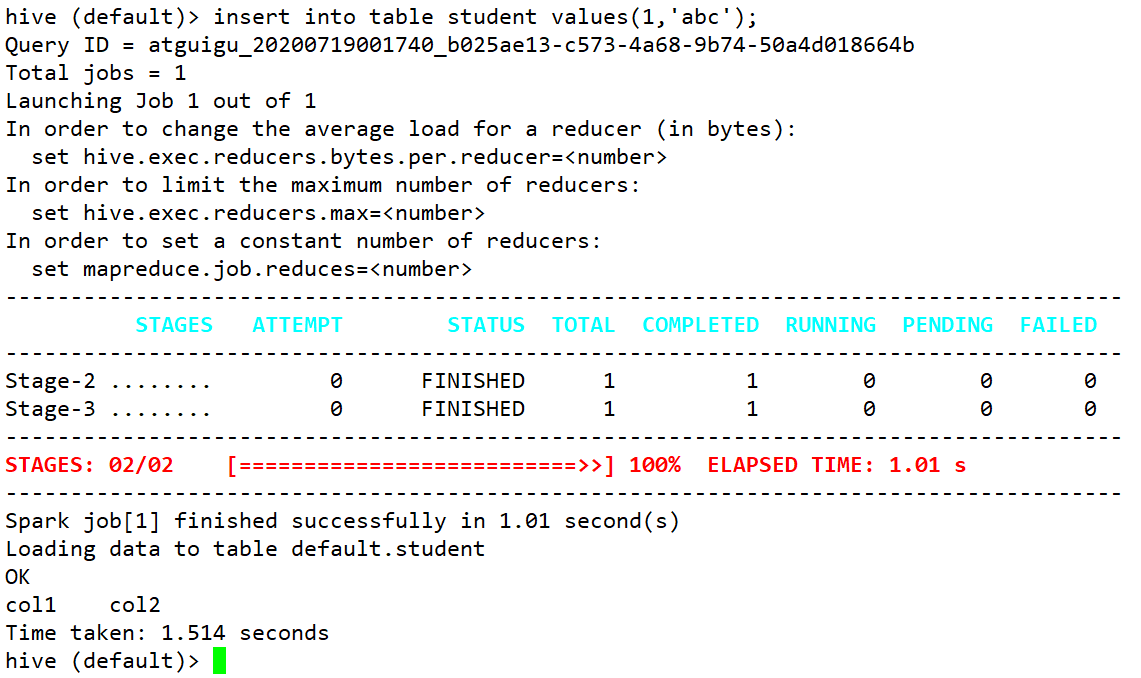
（2）创建一张测试表

hive (default)> create table student(id int, name string);

（3）通过insert测试效果

hive (default)> insert into table student values(1,'abc');

若结果如下，则说明配置成功



### 12.4 Yarn容量调度器并发度问题演示

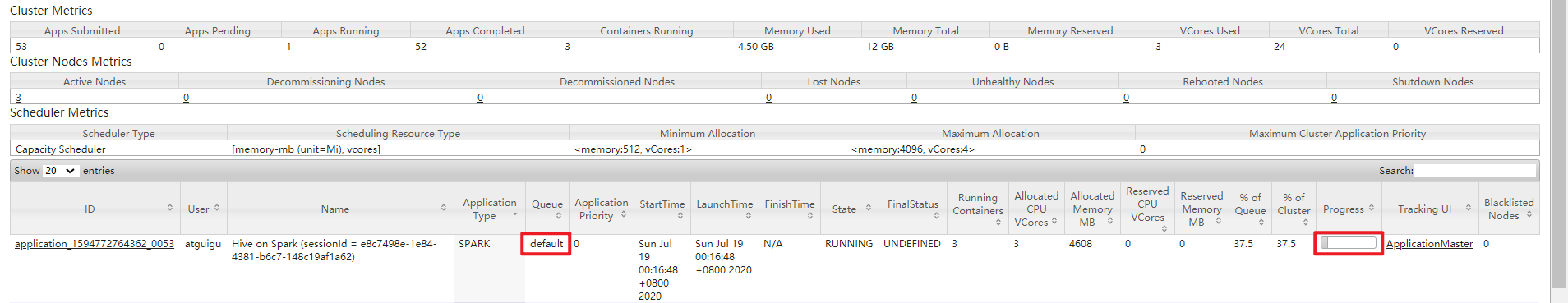
Yarn默认调度器为Capacity Scheduler（容量调度器），且默认只有一个队列——default。如果队列中执行第一个任务资源不够，就不会再执行第二个任务，一直等到第一个任务执行完毕。

（1）启动1个hive客户端，执行以下插入数据的sql语句。

hive (default)> insert into table student values(1,'abc');

执行该语句，hive会初始化一个Spark Session，用以执行hive on spark任务。由于未指定队列，故该Spark Session默认占用使用的就是default队列，且会一直占用该队列，直到退出hive客户端。

可访问ResourceManager的web页面查看相关信息。

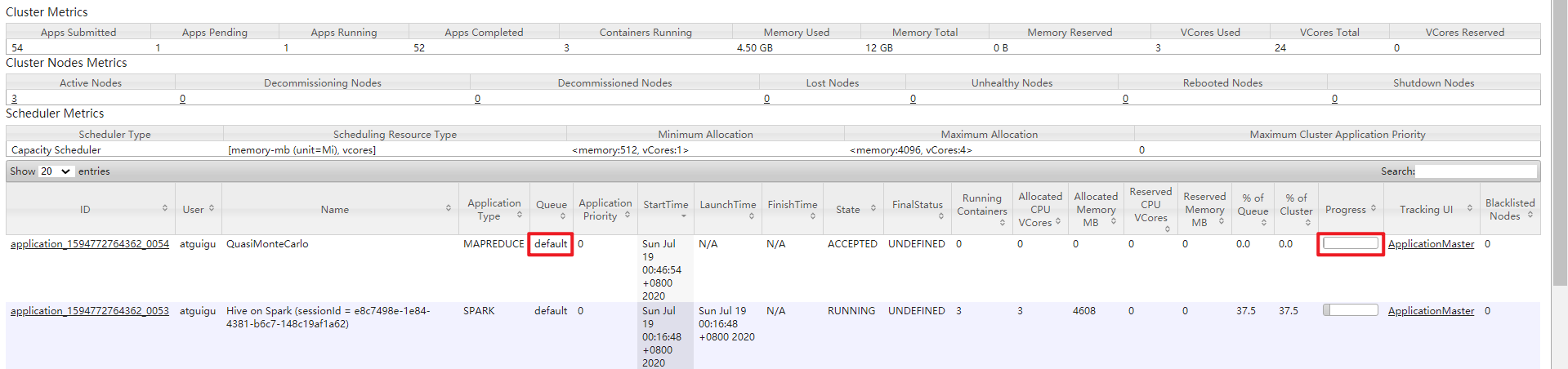


（2）在hive客户端开启的状态下，提交一个MR。

[xiao@hadoop102 ~]$ hadoop jar /opt/module/hadoop-3.1.3/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.1.3.jar pi 1 1

MR任务同样未指定队列，所以其默认也提交到了default队列，由于容量调度器单个队列的并行度为1。故后提交的MR任务会一直等待，不能开始执行。

ResourceManager的web页面如下：



（3）容量调度器default队列中，同一时间只有一个任务执行，并发度低，如何解决呢？

方案一：增加ApplicationMaster资源比例，进而提高运行app数量。

方案二：创建多队列，比如增加一个hive队列。

### 12.5 增加ApplicationMaster资源比例

针对容量调度器并发度低的问题，考虑调整yarn.scheduler.capacity.maximum-am-resource-percent该参数。默认值是0.1，表示集群上AM最多可使用的资源比例，目的为限制过多的app数量。

（1）在hadoop102的/opt/module/hadoop-3.1.3/etc/Hadoop/capacity-scheduler.xml文件中修改如下参数值

[xiao@hadoop102 hadoop]$ vim capacity-scheduler.xml

<property>

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

<value>0.5</value>

<description>

集群中用于运行应用程序ApplicationMaster的资源比例上限，

该参数通常用于限制处于活动状态的应用程序数目。该参数类型为浮点型，

默认是0.1，表示10%。所有队列的ApplicationMaster资源比例上限可通过参数

yarn.scheduler.capacity.maximum-am-resource-percent设置，而单个队列可通过参数yarn.scheduler.capacity.<queue-path>.maximum-am-resource-percent设置适合自己的值。

</description>

</property>

（2）分发capacity-scheduler.xml配置文件

[xiao@hadoop102 hadoop]$ xsync capacity-scheduler.xml

（3）关闭正在运行的任务，重新启动yarn集群

[xiao@hadoop103 hadoop-3.1.3]$ sbin/stop-yarn.sh

[xiao@hadoop103 hadoop-3.1.3]$ sbin/start-yarn.sh

### 12.6 增加Yarn容量调度器队列

方案二：创建多队列，也可以增加容量调度器的并发度。

在企业里面如何配置多队列：

按照计算引擎创建队列hive、spark、flink

按照业务创建队列：下单、支付、点赞、评论、收藏（用户、活动、优惠相关）

有什么好处？

假如公司来了一个菜鸟，写了一个递归死循环，公司集群资源耗尽，大数据全部瘫痪。

**解耦。**

假如11.11数据量非常大，任务非常多，如果所有任务都参与运行，一定执行不完，怎么办？

**可以支持降级运行。**

下单 √

支付√

点赞X

**1）增加容量调度器队列**

（1）修改容量调度器配置文件

默认Yarn的配置下，容量调度器只有一条default队列。在capacity-scheduler.xml中可以配置多条队列，**修改**以下属性，增加hive队列。

<property>

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

<value>default,hive</value>

<description>

再增加一个hive队列

</description>

</property>

<property>

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

<value>50</value>

<description>

default队列的容量为50%

</description>

</property>

同时为新加队列**添加**必要属性：

<property>

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

<value>50</value>

<description>

hive队列的容量为50%

</description>

</property>

<property>

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

<value>1</value>

<description>

一个用户最多能够获取该队列资源容量的比例，取值0-1

</description>

</property>

<property>

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

<value>80</value>

<description>

hive队列的最大容量（自己队列资源不够，可以使用其他队列资源上限）

</description>

</property>

<property>

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

<value>RUNNING</value>

<description>

开启hive队列运行，不设置队列不能使用

</description>

</property>

<property>

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

<value>\*</value>

<description>

访问控制，控制谁可以将任务提交到该队列,\*表示任何人

</description>

</property>

<property>

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

<value>\*</value>

<description>

访问控制，控制谁可以管理(包括提交和取消)该队列的任务，\*表示任何人

</description>

</property>

<property>

<name>yarn.scheduler.capacity.root.hive.acl\_application\_max\_priority</name>

<value>\*</value>

<description>

指定哪个用户可以提交配置任务优先级

</description>

</property>

<property>

<name>yarn.scheduler.capacity.root.hive.maximum-application-lifetime</name>

<value>-1</value>

<description>

hive队列中任务的最大生命时长，以秒为单位。任何小于或等于零的值将被视为禁用。

</description>

</property>

<property>

<name>yarn.scheduler.capacity.root.hive.default-application-lifetime</name>

<value>-1</value>

<description>

hive队列中任务的默认生命时长，以秒为单位。任何小于或等于零的值将被视为禁用。

</description>

</property>

（2）分发配置文件

[xiao@hadoop102 ~]$ xsync /opt/module/hadoop-3.1.3/etc/hadoop/capacity-scheduler.xml

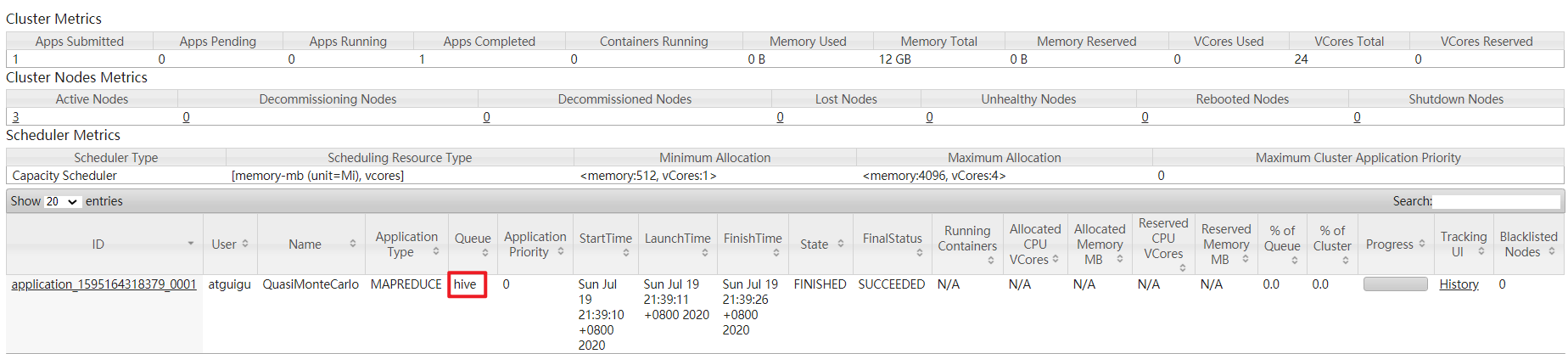
（3）重启Hadoop集群

**2）测试新队列**

（1）提交一个MR任务，并指定队列为hive

[xiao@hadoop102 ~]$ hadoop jar /opt/module/hadoop-3.1.3/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.1.3.jar pi -Dmapreduce.job.queuename=hive 1 1

（2）查看ResourceManager的web页面，观察任务被提交到的队列



### 12.7 创建数据库

1）启动hive

[xiao@hadoop102 hive]$ bin/hive

2）显示数据库

hive (default)> show databases;

3）创建数据库

hive (default)> create database gmall;

4）使用数据库

hive (default)> use gmall;