### 前提准备:

软件：/data/apps/

数据： /data/hdfs-data

机器：172.16.x.x  172.16.x.x 172.16.x.x

有sudo权限的用户

hdpusr

hdpusr

### 集群规划：

172.16.xx.xx node1 dn nm nn hive spark

172.16.xx.xx node2 dn nm snm rm spark

172.16.xx.xx node3 dn nm mysql spark

172.16.xx.xx node4 dn nm spark

172.16.xx.xx node5 dn nm spark

172.16.xx.xx node6 dn nm spark

在etc/hosts文件中不要将127.0.0.1与hostname做映射 否则后面会出现NM去连接NN的9000端口时出问题

### 版本信息:

jdk1.8

scala2.11.8

hadoop2.6.4

spark2.1.1&2.4

hive-2.0.1

mysql-5.7.24

### 服务器环境配置:

修改主机名：

1. uname -a 查看hostname

2. hostname newname 修改下，让hostname立刻生效。

3. vi /etc/hosts 修改原hostname为 newname

4. vi /etc/sysconfig/network 修改原hostname为 newname

5. reboot重启后也生效

centos7可以直接使用hostnamectl命令修改:

1,hostnamectl set-hostname newname

2,修改/etc/hosts 文件中旧name

建立免密登录:

cd /home/username/.ssh

ssh-keygen -t rsa

scp -r id\_rsa.pub username@node3:/home/username /.ssh/authorized\_keys\_from\_node1 #将id\_rsa.pub发往被登录的服务器

cat authorized\_keys\_from\_node1 >> authorized\_keys #在被登录的服务器合并秘钥

失败->检测.ssh目录以及文件权限 执行：chmod 700 .ssh/ chmod 700 .ssh/\*

ssh localhost的配置上面的操作一致将自己生成的id\_rsa.pub追加入authorized\_keys文件中即可

环境变量配置:

export JAVA\_HOME=/data/apps/jdk

export SCALA\_HOME=/data/apps/scala

export HADOOP\_HOME=/data/apps/hadoop

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

### hadoop集群安装：

参数配置:

hadoop-env.sh:

export HADOOP\_LOG\_DIR=/data/logs/hadoop-log/

core-site.xml:

<property>

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

<value>/data/apps/hadoop/tmp/</value>

<description>A base for other temporary directories.</description>

</property>

<property>

<name>fs.defaultFS</name>

<value>hdfs://node1</value>

</property>

<!-- file system properties -->

<property>

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

<value>hdfs://node1:9000</value>

</property>

hdfs-site.xml:

<property>

<name>dfs.replication</name>

<value>2</value>

</property>

<property>

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

<value>node2:50090</value>

</property>

<property>

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

    <value>file://data/hdfs-data/</value>

</property>

mapred-site.xml:

<!-- 指定mr运行在yarn上 -->

<property>

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

<value>yarn</value>

</property>

yarn-site.xml:

<property>

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

<value>node2</value>

</property>

<property>

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

<value>mapreduce\_shuffle</value>

</property>

<property>

<name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>

<value>org.apache.hadoop.mapred.ShuffleHandler</value>

</property>

<property>

<description>Whether to enable log aggregation</description>

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

<value>true</value>

</property>

<property>

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

<value>22528</value>

<description>Usable memory</description>

</property>

<property>

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

<value>8</value>

<description>Number of virtual CPUs that can be used**(逻辑CPU的总数**)</description>

</property>

<property>

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

<value>org.apache.hadoop.yarn.server.resourcemanager.scheduler.fair.FairScheduler</value>

</property>

<property>

<name>yarn.scheduler.fair.allocation.file</name>

<value>/data/apps/hadoop/etc/hadoop/fair-scheduler.xml</value>

</property>

<property>

<name>yarn.scheduler.fair.preemption</name>

<value>true</value>

</property>

<property>

<name>yarn.scheduler.fair.user-as-default-queue</name>

<value>true</value>

<description>default is True</description>

</property>

<property>

<name>yarn.scheduler.fair.allow-undeclared-pools</name>

<value>false</value>

<description>default is True</description>

</property>

yarn-env.sh

export YARN\_LOG\_DIR=/data/logs/hadoop-log/

启动hadoop：

hadoop namenode -format

Namenode和ResourceManger如果不是同一台机器，不能在NameNode上启动 yarn，应该在ResouceManager所在的机器上启动yarn;

刷新yarn队列的配额:

yarn rmadmin –refreshQueues

### spark on yarn集群安装

spark-defaults.conf：

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

spark.driver.memory 1g

spark.executor.extraJavaOptions -XX:+PrintGCDetails -Dkey=value -Dnumbers="one two three"

spark-env.sh：

export JAVA\_HOME==/data/apps/jdk

export SCALA\_HOME=/data/apps/scala

export HADOOP\_HOME=/data/apps/hadoop

export HADOOP\_CONF\_DIR=/data/apps/hadoop/etc/hadoop

export SPARK\_MASTER\_PORT=7077

export SPARK\_EXECUTOR\_CORES=2

export SPARK\_EXECUTOR\_MEMORY=1024m

export SPARK\_EXECUTOR\_INSTANCES=1

log4j.properties:

spark 日志级别设置为WARN

/data/apps/spark/conf/log4j.properties

### yarn做cpu和gpu任务隔离 (使用capacity-scheduler + nodeLabel实现)

添加label：

yarn rmadmin -addToClusterNodeLabels gpu

node添加label:

yarn rmadmin -replaceLabelsOnNode node4,gpu

yarn rmadmin -replaceLabelsOnNode node2:28128,cpu(如果不添加端口则会将label添加到节点的所有nodeManager上)

查看标签状态:

yarn node -status node1:14713

yarn rmadmin -refreshQueues

执行脚本:

yarn rmadmin -addToClusterNodeLabels gpu

yarn rmadmin -replaceLabelsOnNode node4,gpu

yarn rmadmin -replaceLabelsOnNode node5,gpu

yarn rmadmin -replaceLabelsOnNode node6,gpu

配置:

<configuration>

<!-- 父队列root的相关配置 -->

<property>

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

<value>default,gpu\_queue</value>

</property>

<property>

<name>yarn.scheduler.capacity.root.accessible-node-labels.gpu.capacity</name>

<value>100</value>

<description>root队列对gpu标签节点可用的百分比</description>

</property>

<!-- 队列default的相关配置 -->

<property>

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

<value>50</value>

</property>

<!-- 队列gpu\_queue的相关配置 -->

<property>

<name>yarn.scheduler.capacity.root.gpu\_queue.capacity</name>

<value>50</value>

<description>gpu\_queue队列可用root队列资源的百分比</description>

</property>

<property>

<name>yarn.scheduler.capacity.root.gpu\_queue.maximum-capacity</name>

<value>100</value>

<description>gpu\_queue队列资源使用上限</description>

</property>

<property>

<name>yarn.scheduler.capacity.root.gpu\_queue.accessible-node-labels</name>

<value>gpu</value>

<description>gpu\_queue队列应用可用的节点标签</description>

</property>

<property>

<name>yarn.scheduler.capacity.root.gpu\_queue.default-node-label-expression</name>

<value>gpu</value>

<description>gpu\_queue队列应用默认节点标签</description>

</property>

<property>

<name>yarn.scheduler.capacity.root.gpu\_queue.accessible-node-labels.gpu.capacity</name>

<description>gpu\_queue队列对gpu标签节点可用的百分比</description>

<value>100</value>

</property>

</configuration>

### 查看web界面

在node2机器上执行:

w3m http://node2:8088

w3m http://node1:50070

### yarn的rest api：

--集群信息

http://node2:8088/ws/v1/cluster/info

--资源信息

http://node2:8088/ws/v1/cluster/metrics

--调度

http://node2:8088/ws/v1/cluster/scheduler

--应用程序

http://node2:8088/ws/v1/cluster/apps

--应用程序状态

http://node2:8088/ws/v1/cluster/appstatistics?states=accepted,running,finished&applicationTypes=mapreduce

--节点信息

http://node2:8088/ws/v1/cluster/nodes

curl --compressed -H "Accept: application/json" -X GET "http://node2:8088/ws/v1/cluster/scheduler"

### hive安装

hive-env.sh

HADOOP\_HOME=/data/apps/hadoop

export HIVE\_CONF\_DIR=/data/apps/hive/conf

export HIVE\_AUX\_JARS\_PATH=/data/apps/hadoop/lib

hive-site配置

<property>

<name>hive.metastore.uris</name>

<value>thrift://node1:9083</value>

</property>

<property>

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

<value>/tmp/hive/java</value>

</property>

<property>

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

<value>username</value>

</property>

<property>

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

<value>com.mysql.jdbc.Driver</value>

</property>

<property>

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

<value>jdbc:mysql://node3:3306/hive?createDatabaseIfNotExist=true&amp;useSSL=true</value>

</property>

<property>

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

<value>root</value>

</property>

<property>

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

<value>root</value>

</property>

添加Mysql依赖包:

cp ~/mysql-connector-java-5.1.39.jar /data/apps/hive/lib/

对数据库进行初始化

schematool -initSchema -dbType mysql

#启动metastore:

./bin/hive --service metastore 1>/dev/null 2>&1 &

#将mysql驱动包和hive-site.xml复制到spark的jars和conf目录下

scp -r mysql-connector-java-5.1.39.jar username@node2:/data/apps/spark/jars/

scp -r conf/hive-site.xml username@node2:/data/apps/spark/conf/

### mysql安装

卸载系统自带的Mariadb

rpm -qa|grep mariadb //查询出已安装的mariadb

rpm -e --nodeps 文件名 //卸载 ， 文件名为使用rpm -qa|grep mariadb 命令查出的所有文件

rm /etc/my.cnf

1.解压：

mysql-5.7.24-linux-glibc2.12-x86\_64.tar

继续解压：

tar -zxvf mysql-5.7.24-linux-glibc2.12-x86\_64.tar.gz

2.安装和初始化MySQL数据库：

bin/mysqld --initialize --user=username --basedir=/data/apps/mysql/ --datadir=/data/apps/mysql/data/

在mysql目录下拷贝mysql.server文件到 /etc/init.d/mysqld

执行命令：cp -a ./support-files/mysql.server /etc/init.d/mysqld

编辑/etc/init.d/mysqld 添加数据库安装目录，和数据存放的地方

basedir这个目录是数据库的根目录 也就是安装目录(注意:是安装目录)

datadir这个目录是数据存放的地方

vim /etc/init.d/mysqld

删除系统默认的MySQL配置文件/etc/my.cnf

rm -rf /etc/my.cnf

3.初始化MySQL

./mysqld\_safe --user=username &

4.启动MySQL

/etc/init.d/mysqld restart

5.初始化密码设置

查看初始化密码:

在bin执行以下命令:cat /root/.mysql\_secret

\_yPyalNtgdig

修改初始化密码：

ERROR 1045 (28000): Access denied for user 'root'@'localhost' (using password: YES)

/etc/init.d/mysqld stop

./mysqld\_safe --user=username --skip-grant-tables --skip-networking &

mysql -u root mysql

UPDATE user SET authentication\_string=PASSWORD('root') where USER='root';

flush privileges;

ERROR 1820 (HY000): You must reset your password using ALTER USER statement before executing this statement.

ALTER USER 'root'@'localhost' IDENTIFIED BY 'root';

6.添加远程访问

use mysql;

update user set host = '%' where user = 'root';

启动hive：

./bin/hive --service metastore 1>/dev/null 2>&1 &

### Livy搭建

配置livy-env.sh

JAVA\_HOME=/data/apps/jdk

HADOOP\_CONF\_DIR=/data/apps/hadoop

SPARK\_HOME=/data/apps/spark

配置livy.conf:

# What host address to start the server on. By default, Livy will bind to all network interfaces.

livy.server.host = node2

# What port to start the server on.

livy.server.port = 8998

# What spark master Livy sessions should use.

livy.spark.master = yarn

# What spark deploy mode Livy sessions should use.

livy.spark.deploy-mode =client

开启失败重试机制:

livy.server.recovery.mode = recovery

livy.server.recovery.state-store = filesystem

livy.server.recovery.state-store.url = hdfs://node1:9000/data

开启spark sessin但出现Table or view not found:异常:

### 系统资源查看:

查看物理cpu个数：

cat /proc/cpuinfo | grep "physical id" | sort | uniq|wc -l

查看每个cpu的核数：

cat /proc/cpuinfo | grep "cores"|uniq

查看逻辑cpu的个数：

cat /proc/cpuinfo | grep "processor" |wc -l

逻辑CPU的总数=物理CPU的数量 \* 每个物理CPU上的核数 \* 超线程数