本文旨在描述搭建一个大数据实验室使用的大数据计算平台，并详细记录了整个平台的搭建过程和遇到的问题。本文会从两个大方面展开：其一是介绍了整个平台从需求到调研再到最终平台的架构设计思路和我个人的一些浅薄理解；其二记录以平台架构为指导的平台搭建过程。虽然我是个初学者，但是在学习的同时不忘思考如何从大数据平台架构师的角度思考如何才能搭建出一个更好的平台，我觉得这一点是很重要的。

## 大数据计算平台设计

### 平台框架

首先我们应该知道构建一个大数据计算平台，应该具备的核心技术有：存储中心、集群管理、计算框架、数据来源等。其中对各个技术选择使用的工具作如下介绍：

1. 存储中心：存储当然选择的是目前最好的Hadoop HDFS，将来可能还会结合目前正在发展的Tachyon，实现磁盘分布式文件系统和内存分布式文件系统的结合。
2. 集群管理：目前两个比较好的集群管理软件都是Apache的子项目：Mesos和Hadoop的Yarn。鉴于我已经使用Hadoop的HDFS，而且Yarn也不比Mesos差，所以就选择Yarn。
3. 计算框架：顺应时代潮流，放弃MapReduce直接使用Spark，但是我们的集群管理使用的是Yarn，所以还是可以同时使用MapReduce、Spark及其他的计算框架的，这样就提高了系统的可扩展性，不然就没有必要使用Yarn了，Spark的Standalone模式完全可以支撑一个小型集群了。多说一句，我觉得要深入了解Spark还是要好好阅读以下AMPlab这篇经典的论文--《Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing》。
4. 数据来源：对数据来源方式还没有做过研究，但是仅仅对于数据流的实时处理，数据来源是一个绕不过去的问题，其中Spark自己内嵌的Spark SQL是个很不错的工具。先占个坑，以后补全。

以上就是我们要选择的系统，还有一点值得一提，HDFS的NameNode节点、Yarn的ResourceManager节点和Spark的Master节点都存在SPOF(单点故障)问题，需要结合ZooKeeper来实现高可靠。其中HDFS使用的是QJM特性和ZooKeeper，Yarn已经内嵌了ZooKeeper的ActiveStandbyElector(官方：Note that, there is no need to run a separate ZKFC daemon as is the case for HDFS because ActiveStandbyElector embedded in RMs acts as a failure detector and a leader elector instead of a separate ZKFC deamon.)，Spark也可以使用Zookeeper，但是因为我们是基于yarn模式的，而不是spark自身的standalone模式，所以暂不需要，具体内容后面搭建时还会讲到。

综上，我们就可以得出平台的框架，其实在展示我的框架之前，有必要给出AMP实验室官网上的一张图：BDAS(Berkeley Data Analytics Stack，伯克利数据分析技术栈)，因为这张图涉及的技术已经很全面了，我也是借鉴这张图来搭建平台的，希望以后可以将这个平台迭代成这张图的样子。

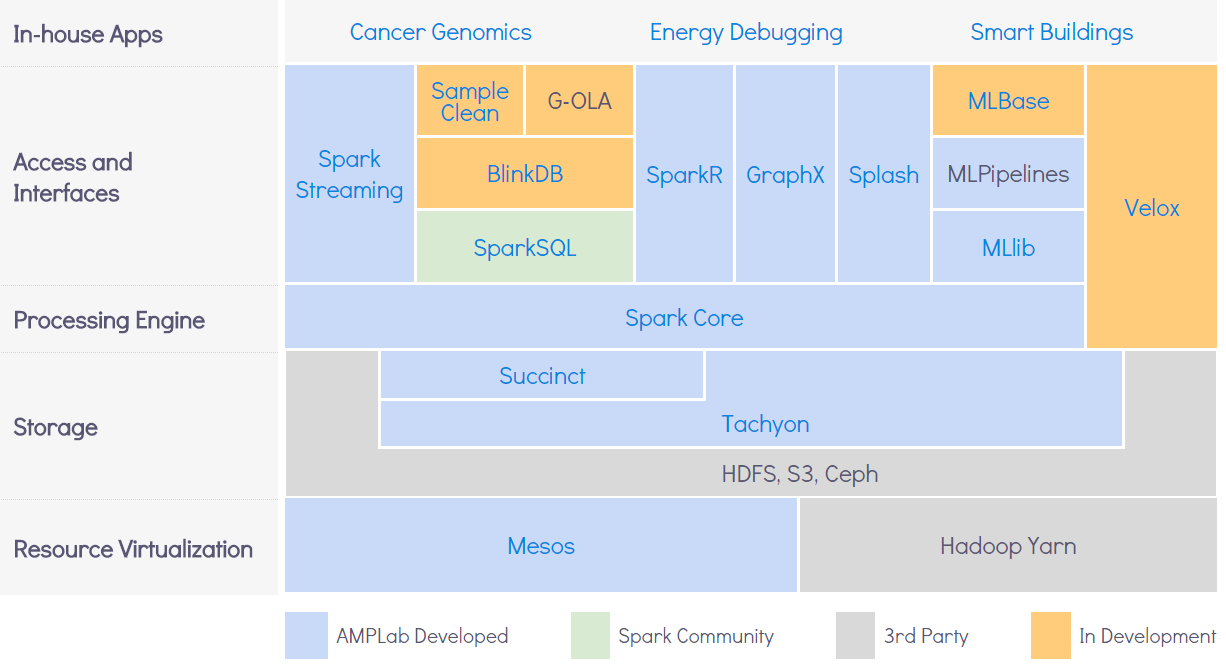


图1-BDAS

下面就是目前要搭建的平台框架：

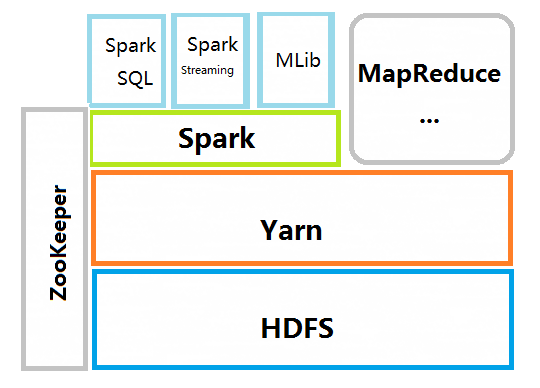


图2-平台设计框架

### 平台部署的问题

系统部署方面，我觉得最值得说的就是自动化部署了，虽然目前这里所说的自动化不能完全称得上自动化部署(如果将来可能和Docker结合的话，会实现真正的自动化)，但是要想作为一个优秀的运维工程师，必须时刻记得把重复的体力劳动交给计算机去处理(脚本程序)，而且我看到网上的教程基本上忽略这个，部署过程中都是手动配置，我以为即使集群再小也要培养自己使用脚本提高效率的能力，所以在部署之前我觉得应该说明一下在以下情况会用到脚本(先假设所有的主机都预装了openssh-server，如果没有这个，谈什么自动化？)：

1. 自动构建ssh无密码登陆。
2. 自动安装软件，如JDK和Scala。
3. 自动配置环境变量以及每个节点保存其他节点的名字和IP的对应关系。
4. 批量复制在主机上配置好的各个系统的文件或同步个别已修改的文件。

因为时间原因，脚本写的很粗糙，而且有很多冗余代码，希望读者在此基础上改进使用。

## 平台搭建过程

### 搭建环境说明

1. 五台Ubuntu 14.04台式机
2. 网络环境：网络是实验室的局域网，所以五台实验机并非处于私有局域网，这样就限制了指定静态IP，从而给后面的扩展带来了一些麻烦，有条件的话建议给集群建一个私有的局域网。
3. 在搭建之前尽量保持系统环境的清洁，避免使用多次配置Hadoop或者Spark的系统。
4. 给每台机子创建相同的用户名，因为Hadoop的控制节点在使用无密码登陆从节点时，会使用你在控制节点上运行Hadoop程序的用户名，如果从节点上没有相同的用户名会发生登陆不上从节点主机。在Ubuntu上，如果你的用户名不是sudoer，建议将其添加到/etc/sudoers文件中，因为我们会使用ssh修改root用户的文件，方法自己上网搜。
5. 所有节点上Hadoop等程序的放置目录必须一样。

### 环境准备

1. 先将各个从节点主机的主机名和IP的对应关系添加到master节点的/etc/hosts文件中。

master@master:~/workspace$ cat /etc/hosts

#127.0.0.1 localhost

115.154.138.7 master.hadoop-cluster master

115.154.138.31 lilei.hadoop-cluster lilei

115.154.138.34 hanmei.hadoop-cluster hanmei

115.154.138.35 lucy.hadoop-cluster lucy

115.154.138.33 lily.hadoop-cluster lily

这里的lilei.hadoop-cluster之类的只是各个主机的主机名，其上的用户都是master。注意，我这里将master.hadoop-cluster自己的主机名映射为他自己的对外IP，而不是环回地址，这在Ubuntu系统上是臭名昭著的。

注意：下面文中还会多处提到master节点和slave节点，这里所说的master节点是指我们操作配置的主机，而不是类似spark的master节点，而slave节点就是被我们远程控制的主机。

1. 自动构建ssh无密码登陆

先在控制主机上生成公钥

master@master:~/workspace$ ssh-keygen -t rsa

然后将ssh创建的公钥id\_rsa.pub的内容复制到各个从主机的ssh环境的authosized\_keys文件中，这里要用到四个脚本：noscp.exp、writeauthkey.exp、remotesudo.exp 和createSshNopasswd.sh。

代码如下：

noscp.exp文件可以在用scp复制文件时不需要输入密码的交互：

master@master:~/workspace$ cat noscp.exp

#!/usr/bin/expect

#./noscp.exp localfile remotefile

#scp file without passsword(no interaction)

if {$argc<4} {

puts stderr "Usage: $argv0 localfile remotefile user passwd"

exit 1

}

set localfile [ lindex $argv 0 ]

set remotefile [ lindex $argv 1 ]

set user [ lindex $argv 2 ]

set pwd [ lindex $argv 3 ]

set timeout 20

spawn scp ${localfile} ${user}@${remotefile}

expect {

"\*yes/no" { send "yes\r"; exp\_continue }

"\*password:" { send "$pwd\r" }

}

expect eof

writeauthkey.exp文件将复制到各主机的id\_rsa.pub文件写到~/.ssh/authorized\_keys文件中：

master@master:~/workspace$ cat writeauthkey.exp

#!/usr/bin/expect

#write master's id\_rsa.pub to slaves

if {$argc<4} {

puts stderr "Usage: $argv0 host remotefile user passwd"

exit 1

}

set host [ lindex $argv 0 ]

set remotefile [ lindex $argv 1 ]

set user [ lindex $argv 2 ]

set pwd [ lindex $argv 3 ]

set timeout 20

spawn ssh ${user}@${host} cat ${remotefile} >> ~/.ssh/authorized\_keys

expect {

"\*yes/no" { send "yes\r"; exp\_continue }

"\*password:" { send "$pwd\r" }

}

expect eof

remotesudo.exp文件可以是用户在使用ssh执行sudo命令时不需要交互输入密码，从而达到自动化到各个主机上执行sudo命令。

master@master:~/workspace$ cat remotesudo.exp

#!/usr/bin/expect

#ssh to slaves executing sudo commands

if {$argc<4} {

puts stderr "Usage: $argv0 command host user passwd"

exit 1

}

set command [ lindex $argv 0 ]

set host [ lindex $argv 1 ]

set user [ lindex $argv 2 ]

set pwd [ lindex $argv 3 ]

set timeout 20

spawn ssh -t ${user}@${host} sudo ${command}

expect {

"\*password\*" { send "$pwd\r" }

}

expect eof

createSshNopasswd.sh使用上面的三个脚本对每个主机循环操作(原谅我将主机的密码设置成这么弱的口令，而且还保存在脚本中，原因是目前的计算集群都是在一个局域网实现的，安全方面没有什么问题，这里偷了个懒，如果是对外提供接口，还是不要这么做)：

master@master:~/workspace$ cat createSshNopasswd.sh

#!/bin/bash

#this script is used to automaticly realize master ssh to slaves no password

file=/etc/hosts

localfile=~/.ssh/id\_rsa.pub

if [ -e "$localfile" ]

then

for var in `grep "hadoop" /etc/hosts | grep -v "master" |sed 's/^.\*hadoop-cluster //g' 2>/dev/null`

do

if grep $var $file 1>/dev/null 2>/dev/null

then

./noscp.exp $localfile ${var}.hadoop-cluster:/home/ master / master 123456

./writeauthkey.exp ${var}.hadoop-cluster /home/master/\_rsa.pub master 123456

./remotesudo.exp "service ssh restart" ${var}.hadoop-cluster master 123456

fi

done

fi

1. 将各个主机的主机名和IP对应关系添加到其他主机/etc/hosts，保持主机彼此认识。

master@master:~/workspace$ cat setHostIpandName.sh

#!/bin/bash

#this scripts is used to set clusters’s hostnames and ips on every host

masterip=115.154.138.7

endfix=adoop-cluster

pwd=123456

file=/etc/hosts

localfile=~/.ssh/id\_rsa.pub

username=master

if [ -e “$localfile” ]

then

for var in `grep “adoop” ${file} | grep –v “master” |sed ‘s/^.\*adoop-cluster //g’ 2>/dev/null`

do

./remotesudo.exp “sed –i ‘\$a\\${masterip} master.${endfix} master’ /etc/hosts” ${var}.${endfix} ${username} ${pwd}

IFS.OLD=$IFS

IFS=$’\n’

for iphost in `grep “adoop” ${file} | grep –v $var | grep –v “master”`

do

./remotesudo.exp “sed –i ‘\$a\\${iphost}’ /etc/hosts” ${var}.${endfix} ${username} ${pwd}

done

IFS=$IFS.OLD

done

fi

1. 使每个主机对其他主机都可以ssh无密码登陆，这一点在实现HA时，很重要，因为你的master节点可能切换到其他slave节点上。

master@master:~/workspace$ cat createSshNopasswdInSlave.sh

#!/bin/bash

#this script is used to automaticly realize master ssh to slaves no password

file=/etc/hosts

localfile=~/.ssh/id\_rsa.pub

commdir=/home/master/workspace

hostname=`cat /etc/hostname`

if [ -e "$localfile" ]

then

for var in `grep "hadoop" /etc/hosts | grep -v ${hostname} |sed 's/^.\*hadoop-cluster //g' 2>/dev/null`

do

if grep $var $file 1>/dev/null 2>/dev/null

then

${commdir}/noscp.exp $localfile ${var}.hadoop-cluster:/home/master/ master 123456

${commdir}/writeauthkey.exp ${var}.hadoop-cluster /home/master/id\_rsa.pub master 123456

${commdir}/remotesudo.exp "service ssh restart" ${var}.hadoop-cluster master 123456

fi

done

fi

上面这个脚本是复制到各个slave节点上运行的，下面的是在master主机上与运行的。

master@master:~/workspace$ cat createSshNopasswdEachOther.sh

#!/bin/bash

#this script is used to automaticly realize master ssh no password

dir=/home/master/workspace

./cpFilesToSlaves.sh ./sshkeyGen.exp ${dir}

./cpFilesToSlaves.sh ./noscp.exp ${dir}

./cpFilesToSlaves.sh ./writeauthkey.exp ${dir}

./cpFilesToSlaves.sh ./remotesudo.exp ${dir}

./cpFilesToSlaves.sh ./createSshNopasswdInSlave.sh ${dir}

hostname=`cat /etc/hostname`

file=/etc/hosts

localfile=~/.ssh/id\_rsa.pub

for var in `grep "hadoop" /etc/hosts | grep -v "master" |sed 's/^.\*hadoop-cluster //g' 2>/dev/null`

do

if grep $var $file 1>/dev/null 2>/dev/null

then

ssh ${var}.hadoop-cluster ${dir}/sshkeyGen.exp

ssh ${var}.hadoop-cluster ${dir}/createSshNopasswdInSlave.sh

fi

done

1. 接下来安装JDK、Scala和配置环境变量，Scala要先在官网下载压缩包，因为Ubuntu的dpkg数据库是Scala 2.9的，spark官方建议下载Scala 2.10以后版本，我的压缩包下载后放在本地的/home/master/workspace/目录下。

master@master:~/workspace$ cat remotesudoNoWait.exp

#!/usr/bin/expect

#ssh to slaves executing sudo commands wait for long time(take much time when install jdk)

if {$argc<4} {

puts stderr “Usage: $argv0 command host user passwd”

exit 1

}

set command [ lindex $argv 0 ]

set host [ lindex $argv 1 ]

set user [ lindex $argv 2 ]

set pwd [ lindex $argv 3 ]

set timeout 3600

spawn ssh –t ${user}@${host} sudo ${command}

expect {

“\*password\*” { send “$pwd\r” }

}

expect eof

master@master:~/workspace$ cat installDependenciesAndSetEnv.sh

#!/bin/bash

#this scripts is used to intall JDK Scala and set JAVA\_HOME

username=master

endfix=hadoop-cluster

pwd=123456

file=/etc/hosts

java\_home=/usr/lib/jvm/java-1.7.0-openjdk-amd64

scala=/home/master/workspace/scala-2.11.7.tgz

scala\_dir=/home/master/workspace

if [ -e "$file" ]

then

for var in `grep "hadoop" ${file} | grep -v "master" |sed 's/^.\*hadoop-cluster //g' 2>/dev/null`

do

./remotesudoNoWait.exp "apt-get -y install openjdk-7-jre" ${var}.${endfix} ${username} ${pwd}

./remotesudoNoWait.exp "apt-get -y install expect" ${var}.${endfix} ${username} ${pwd}

ssh ${var}.${endfix} grep "JAVA\_HOME" /etc/profile

status=$?

if [ $status -ne 0 ]

then

./remotesudoNoWait.exp "sed -i '\$a\\export JAVA\_HOME=$java\_home' /etc/profile" ${var}.${endfix} ${username} ${pwd}

fi

ssh ${var}.${endfix} which scala

if [ $? -ne 0 ]

then

scp -r $scala master@${var}.${endfix}:${scala\_dir}

ssh master@${var}.${endfix} tar -xzvf ${scala} -C ${scala\_dir}/

./remotesudo.exp "ln -s ${scala\_dir}/scala-2.11.7/bin/scala /usr/bin/" ${var}.${endfix} ${username} ${pwd}

fi

done

fi

### Hadoop HDFS HA和Yarn HA搭建

从官网下载Hadoop和ZooKeeper编译好的文件，解压到一开始规划好的目录，我的目录结构为：工作目录为/home/master/workspace，脚本和Hadoop文件都放在这个目录下，ZooKeeper和Spark放在Hadoop目录下新建的app目录下：

master@master:~/workspace$ ls

noscp.exp

clearTmp.sh remotesudo.exp

cpFilesToSlaves.sh remotesudoNoWait.exp

cpHadoopProjectToSlaves.sh scala-2.11.7

createSshNopasswdEachOther.sh scala-2.11.7.tgz

createSshNopasswdInSlave.sh setEnv.sh

createSshNopasswd.sh setHostIpandName.sh

delFiles.sh sshkey.exp

eclipse sshkeyGen.exp

hadoop-2.7.1 startJournalNode.sh

stopOrstartZk.sh test.sh

installDependenciesAndSetEnv.sh writeauthkey.exp

master@master:~/workspace$ cd adoop-2.7.1

master@master:~/workspace/hadoop-2.7.1$ ls

app data include libexec logs README.txt share

bin etc lib LICENSE.txt NOTICE.txt sbin

五台主机的布置如下：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | master.hadoop-cluster | lilei.hadoop-cluster | hanmei.hadoop-cluster | lily.hadoop-cluster | lucy.hadoop-cluster |
| NameNode | Y | Y | N | N | N |
| DataNode | N | N | Y | Y | Y |
| JournalNode | Y | Y | Y | N | N |
| ZooKeeper | Y | Y | Y | N | N |
| Master(Standalone) | Y | Y | N | N | N |
| Worker(Standalone) | N | N | Y | Y | Y |

为什么是两个NameNode和三个JournalNode的原因，看这里：

<http://hadoop.apache.org/docs/r2.7.1/hadoop-project-dist/hadoop-hdfs/HDFSHighAvailabilityWithQJM.html#Purpose>

ZooKeeper数目应保持奇数，原因看这里：

<http://cailin.iteye.com/blog/2014486>。

上面也列出了spark的standalone模式下的布置情况，但我这里不会配置spark的HA，因为最终我们的spark是要运行在yarn上。

我们这里当然要实现HA自动故障恢复(automatic failover)，在配置ZooKeeper之前还是建议将其实现原理好好熟悉下，不然配置中出现的一些异常都不知道如何下手解决。

下面是各个配置文件的配置内容：

1. ZooKeeper的配置
2. 前面提过ZooKeeper已经解压到了Hadoop-2.7.1的app目录下。将zoo\_sample.cfg文件复制一份命名为zoo.cfg， 修改其中的配置：

tickTime=2000

initLimit=10

syncLimit=5

dataDir=/home/master/workspace/hadoop-2.7.1/app/zookeeper-3.4.6/zkdata

dataLogDir=/home/master/workspace/hadoop-2.7.1/app/zookeeper-3.4.6/zkdatalog

clientPort=2181

server.1=master.hadoop-cluster:2888:3888

server.2=lilei.hadoop-cluster:2888:3888

server.3=hanmei.hadoop-cluster:2888:3888

其中想要指定dataDir和dataLogDir的话，需要自己手动创建目录；sever1、server2和server3即为ZooKeeper集群的所有节点。

1. 进入zkdata文件夹，创建文件myid，填入1。这里写入的1，是在zoo.cfg文本中的server.1中的1。当我们把所有文件都配置完毕，将整个Hadoop工程复制各个主机上时，再修改每台机器中对应的myid文件，lilei.hadoop-cluster中的myid写入2，hanmei.hadoop-cluster中的myid写入3。有更多的节点，安照zoo.cfg的配置，依此写入相应的数字。zkdatalog文件夹，是为了指定zookeeper产生日志指定相应的路径。
2. Hadoop HDFS和HA的配置

共修改6个文件，都位于/home/master/workspace/hadoop-2.7.1/etc/hadoop/目录下：hadoop-env.sh、core-site.xml、hdfs-site.xml、mapred-site.xml、 yarn-site.xml和slaves。

1. hadoop-env.sh文件

添加JDK环境变量

export JAVA\_HOME=/usr/lib/jvm/java-1.7.0-openjdk-amd64

1. core-site.xml文件

<property>

<name>fs.defaultFS</name>

<value>hdfs://mycluster</value> <!—注意，这里的配置和无HA不同mycluster为我定义的HDFS集群名字，这个会在hdfs.site.xml文件中定义 -->

</property>

<property>

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

<value>/home/${user.name}/workspace/hadoop-2.7.1/data/tmp</value>

</property>

<property>

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

<value>master.hadoop-cluster</value>

</property>

<property>

<name>ha.zookeeper.quorum</name> <!-- 这里是ZooKeeper集群的地址和端口 -->

<value>master.hadoop-cluster:2181,lilei.hadoop-cluster:2181,hanmei.hadoop-cluster:2181</value>

</property>

1. hdfs-site.xml文件

<property>

<name>dfs.replication</name>

<value>3</value>

</property>

<property>

<name>dfs.permissions</name>

<value>false</value>

</property>

<property>

<name>dfs.permissions.enabled</name>

<value>false</value>

</property>

<property>

<name>dfs.nameservices</name> <!--给hdfs集群起名字,也就是在b. core-site.xml文件中环境变量fs.defaultFS用到的 -->

<value>mycluster</value>

</property>

<property>

<name>dfs.ha.namenodes.mycluster</name> <!--指定nameservices是mycluster时的namenode有哪些，这里的值也是逻辑名称，名字随便起，相互不重复即可 -->

<value>hadoop1,hadoop2</value>

</property>

<property>

<name>dfs.namenode.http-address.mycluster.hadoop1</name>

<value>master.hadoop-cluster:50070</value> <!--指定hadoop1的http地址-->

</property>

<property>

<name>dfs.namenode.rpc-address.mycluster.hadoop1</name>

<value>master.hadoop-cluster:9000</value> <!--指定hadoop1的RPC地址-->

</property>

<property>

<name>dfs.namenode.servicerpc-address.mycluster.hadoop1</name>

<value>master.hadoop-cluster:53310</value>

</property>

<property>

<name>dfs.namenode.http-address.mycluster.hadoop2</name>

<value>lilei.hadoop-cluster:50070</value> <!--如hadoop1，每个被指定到dfs.ha.namenodes.mycluster中的都要配置 -->

</property>

<property>

<name>dfs.namenode.rpc-address.mycluster.hadoop2</name>

<value>lilei.hadoop-cluster:9000</value>

</property>

<property>

<name>dfs.namenode.servicerpc-address.mycluster.hadoop2</name>

<value>lilei.hadoop-cluster:53310</value>

</property>

<property>

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

<value>true</value> <!--指定mycluster是否启动自动故障恢复-->

</property>

<property>

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

<value>qjournal://master.hadoop-cluster:8485;lilei.hadoop-cluster:8485;hanmei.hadoop-cluster:8485/mycluster</value> <!--指定mycluster的两个NameNode共享edits文件目录时，使用的JournalNode集群信息 -->

</property>

<property>

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

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

</property> <!--指定cluster1出故障时，哪个实现类负责执行故障切换 -->

<property>

<name>dfs.journalnode.edits.dir</name> <!--指定JournalNode集群在对NameNode的目录进行共享时，自己存储数据的磁盘路径。/data/tmp/路径是自己创建，在后面的配置中会出现，journal是启动journalnode自动生成 -->

<value>/home/${user.name}/workspace/hadoop-2.7.1/data/tmp/journal</value>

</property>

<property>

<name>dfs.ha.fencing.methods</name> <!-- 需要NameNode切换时，使用ssh方式进行操作 -->

<value>sshfence</value>

</property>

<property>

<name>dfs.ha.fencing.ssh.private-key-files</name> <!—需要使用ssh进行故障切换时，使用ssh通信时用的密钥存储的位置 -->

<value>/home/${user.name}/.ssh/id\_rsa</value>

</property>

<property>

<name>dfs.ha.fencing.ssh.connect-timeout</name>

<value>10000</value> <!-- HA连接超时时间 -->

</property>

<property>

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

<value>/home/${user.name}/workspace/hadoop-2.7.1/data/name/</value>

</property>

<property>

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

<value>/home/${user.name}/workspace/hadoop-2.7.1/data/edits/</value>

</property>

<property>

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

<value>/home/${user.name}/workspace/hadoop-2.7.1/data/data/</value>

</property>

1. mapred-site.xml文件

配置这个文件主要原因是为了运行Hadoop自带的mapreduce示例程序测试HDFS和YARN。

<property>

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

<value>yarn</value>

</property>

1. yarn-site.xml文件

YARN的HA配置类似于HDFS，这里需要强调的是当使用sbin/start-yarn.sh启动系统YANR集群时，系统并不会启动其他节点的RM，而需要你手动启动，也可能是我没有找到配置自动启动的方法吧。

<!-- 开启resourcemanager ha -->

<property>

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

<value>true</value>

</property>

<property>

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

<value>true</value>

</property>

<!-- 开启自动故障恢复 -->

<property>

<name>yarn.resourcemanager.ha.automatic-failover.enabled</name>

<value>true</value>

</property>

<property>

<name>yarn.resourcemanager.ha.automatic-failover.embedded</name>

<value>true</value>

</property>

<!-- 集群名 -->

<property>

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

<value>myYarnCluster</value>

</property>

<!-- 运行resourcemanager节点名 -->

<property>

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

<value>rm1,rm2</value>

</property>

<!-- 运行resourcemanager节点配置，和hdfs类似 -->

<property>

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

<value>master.hadoop-cluster</value>

</property>

<property>

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

<value>lilei.hadoop-cluster</value>

</property>

<property>

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

<value>master.hadoop-cluster:8088</value>

</property>

<property>

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

<value>lilei.hadoop-cluster:8088</value>

</property>

<property>

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

<value>master.hadoop-cluster:2181,lilei.hadoop-cluster:2181,hanmei.hadoop-cluster:2181</value>

</property>

<property>

<name>yarn.client.failover-proxy-provider</name>

<value>org.apache.hadoop.yarn.client.ConfiguredRMFailoverProxyProvider</value>

</property>

<property>

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

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

</property>

<!--下面是一些通用的配置，内存分配之类的-->

<property>

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

<value>1024</value>

</property>

<property>

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

<value>8192</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>

1. slave文件

hanmei.hadoop-cluster

lucy.hadoop-cluster

lily.hadoop-cluster

1. 将工程复制到其他节点

一个简单的脚本：

master@master:~/workspace/hadoop-2.7.1$ cat ../cpHadoopProjectToSlaves.sh

#!/bin/bash

#this script is used to copy the configured hadoop project to slaves

hostfile=/etc/hosts

dir=/home/master/workspace/hadoop-2.7.1

if [ -d "${dir}" ]

then

for user in `grep "hadoop" $hostfile | grep -v "master" |sed 's/^.\*hadoop-cluster //g' 2>/dev/null`

do

if grep $user $hostfile 1>/dev/null 2>/dev/null

then

ssh master@${user}.hadoop-cluster mkdir /home/master/workspace

scp -r $dir master@${user}.hadoop-cluster:/home/master/workspace/

fi

done

fi

1. 登陆到ZooKeeper的其他节点，修改前面我们配置ZooKeeper时说到的myid文件。
2. 测试
3. 启动ZooKeeper集群

在master.hadoop-cluster、lilei.hadoop-cluster和hanmei.hadoop-cluster上运行bin/zkServer.sh start，脚本：

master@master:~/workspace/hadoop-2.7.1$ cat ../stopOrstartZk.sh

#!/bin/bash

dir=/home/master/workspace/hadoop-2.7.1/app/zookeeper-3.4.6/bin/zkServer.sh

if [ $# -ne 1 ]

then

echo "$0: start/stop"

exit 1

fi

command=$1

if [ $command == "stop" ] || [ $command == "start" ]

then

ssh master.hadoop-cluster $dir $command

ssh lilei.hadoop-cluster $dir $command

ssh hanmei.hadoop-cluster $dir $command

fi

运行结果：

master@master:~/workspace/hadoop-2.7.1$ ../stopOrstartZk.sh start

JMX enabled by default

Using config: /home/master/workspace/hadoop-2.7.1/app/zookeeper-3.4.6/bin/../conf/zoo.cfg

Starting zookeeper ... STARTED

JMX enabled by default

Using config: /home/master/workspace/hadoop-2.7.1/app/zookeeper-3.4.6/bin/../conf/zoo.cfg

Starting zookeeper ... STARTED

JMX enabled by default

Using config: /home/master/workspace/hadoop-2.7.1/app/zookeeper-3.4.6/bin/../conf/zoo.cfg

Starting zookeeper ... STARTED

master@master:~/workspace/hadoop-2.7.1$

master@master:~/workspace/hadoop-2.7.1$ jps

31852 QuorumPeerMain

31914 Jps

master@master:~/workspace/hadoop-2.7.1$

master@master:~/workspace/hadoop-2.7.1$ ssh lilei jps

13093 QuorumPeerMain

13169 Jps

master@master:~/workspace/hadoop-2.7.1$

master@master:~/workspace/hadoop-2.7.1$ ssh hammei jps

ssh: Could not resolve hostname hammei: Name or service not known

master@master:~/workspace/hadoop-2.7.1$ ssh hanmei jps

4646 QuorumPeerMain

4716 Jps

出现上面的结果就是运行正常。

1. 验证格式化zookeeper
2. 在master.hadoop-cluster机器上，执行zkCli.sh 终端上会输出一连串的信息。最后结束的信息是:

2015-08-15 17:31:09,068 [myid:] - INFO [main-SendThread(ip6-localhost:2181):ClientCnxn$SendThread@975] - Opening socket connection to server ip6-localhost/0:0:0:0:0:0:0:1:2181. Will not attempt to authenticate using SASL (unknown error)

JLine support is enabled

2015-08-15 17:31:09,071 [myid:] - INFO [main-SendThread(ip6-localhost:2181):ClientCnxn$SendThread@852] - Socket connection established to ip6-localhost/0:0:0:0:0:0:0:1:2181, initiating session

[zk: localhost:2181(CONNECTING) 0] 2015-08-15 17:31:09,135 [myid:] - INFO [main-SendThread(ip6-localhost:2181):ClientCnxn$SendThread@1235] - Session establishment complete on server ip6-localhost/0:0:0:0:0:0:0:1:2181, sessionid = 0x14f30ae054c0000, negotiated timeout = 30000

WATCHER::

WatchedEvent state:SyncConnected type:None path:null

[zk: localhost:2181(CONNECTED) 0]

[zk: localhost:2181(CONNECTED) 0]

出现上面的结果就是运行正常。

1. 格式化zookeeper集群，目的是在ZooKeeper集群上建立HA的相应节点。

正常的输出结果(中间删掉了不必要的输出信息)：

master@master:~/workspace/hadoop-2.7.1$ bin/hdfs zkfc -formatZK

15/08/15 17:34:36 INFO ha.ActiveStandbyElector: Successfully created /hadoop-ha/mycluster in ZK.

15/08/15 17:34:36 INFO zookeeper.ZooKeeper: Session: 0x34f30ae1af00000 closed

15/08/15 17:34:36 INFO zookeeper.ClientCnxn: EventThread shut down

验证，运行zkCli.sh的正常结果：

master@master:~/workspace/hadoop-2.7.1$ ./app/zookeeper-3.4.6/bin/zkCli.sh

WATCHER::

WatchedEvent state:SyncConnected type:None path:null

[zk: localhost:2181(CONNECTED) 0]

[zk: localhost:2181(CONNECTED) 0]

[zk: localhost:2181(CONNECTED) 0] ls /

[rmstore, yarn-leader-election, hadoop-ha, zookeeper]

[zk: localhost:2181(CONNECTED) 1]

1. 手动到各个ZooKeeper节点启动JournalNode进程

脚本：

master@master:~/workspace/hadoop-2.7.1$ cat ../startJournalNode.sh

#!/bin/bash

command="/home/master/workspace/hadoop-2.7.1/sbin/hadoop-daemon.sh start journalnode"

ssh master.hadoop-cluster $command

ssh lilei.hadoop-cluster $command

ssh hanmei.hadoop-cluster $command

运行结果：

master@master:~/workspace/hadoop-2.7.1$ ../startJournalNode.sh

starting journalnode, logging to /home/master/workspace/hadoop-2.7.1/logs/hadoop-master-journalnode-master.hadoop-cluster.out

starting journalnode, logging to /home/master/workspace/hadoop-2.7.1/logs/hadoop-master-journalnode-lilei.out

starting journalnode, logging to /home/master/workspace/hadoop-2.7.1/logs/hadoop-master-journalnode-hanmei.out

master@master:~/workspace/hadoop-2.7.1$

master@master:~/workspace/hadoop-2.7.1$ jps

31852 QuorumPeerMain

5017 Jps

4877 JournalNode

master@master:~/workspace/hadoop-2.7.1$

master@master:~/workspace/hadoop-2.7.1$ ssh lilei jps

13093 QuorumPeerMain

13365 Jps

13271 JournalNode

master@master:~/workspace/hadoop-2.7.1$

master@master:~/workspace/hadoop-2.7.1$ ssh hanmei jps

4646 QuorumPeerMain

4949 Jps

4855 JournalNode

master@master:~/workspace/hadoop-2.7.1$

1. 格式化集群的一个NameNode，正常输出结果：

master@master:~/workspace/hadoop-2.7.1$ bin/hdfs namenode -format -clusterId mycluster

15/08/15 19:15:12 INFO namenode.FSImage: Allocated new BlockPoolId: BP-714597928-115.154.138.7-1439637312009

15/08/15 19:15:12 INFO common.Storage: Storage directory /home/master/workspace/hadoop-2.7.1/data/name has been successfully formatted.

15/08/15 19:15:12 INFO common.Storage: Storage directory /home/master/workspace/hadoop-2.7.1/data/edits has been successfully formatted.

15/08/15 19:15:12 INFO namenode.NNStorageRetentionManager: Going to retain 1 images with txid >= 0

15/08/15 19:15:12 INFO util.ExitUtil: Exiting with status 0

15/08/15 19:15:12 INFO namenode.NameNode: SHUTDOWN\_MSG:

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

SHUTDOWN\_MSG: Shutting down NameNode at master.hadoop-cluster/115.154.138.7

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

master@master:~/workspace/hadoop-2.7.1$

验证：

master@master:~/workspace/hadoop-2.7.1$ sbin/hadoop-daemon.sh start namenode

starting namenode, logging to /home/master/workspace/hadoop-2.7.1/logs/hadoop-master-namenode-master.hadoop-cluster.out

master@master:~/workspace/hadoop-2.7.1$

master@master:~/workspace/hadoop-2.7.1$

master@master:~/workspace/hadoop-2.7.1$ jps

31852 QuorumPeerMain

4877 JournalNode

31223 Jps

31102 NameNode

master@master:~/workspace/hadoop-2.7.1$

1. 启动一个namenode

master@master:~/workspace/hadoop-2.7.1$ sbin/hadoop-daemon.sh start namenode

starting namenode, logging to /home/master/workspace/hadoop-2.7.1/logs/hadoop-master-namenode-master.hadoop-cluster.out

master@master:~/workspace/hadoop-2.7.1$

master@master:~/workspace/hadoop-2.7.1$ jps

31852 QuorumPeerMain

4877 JournalNode

31223 Jps

31102 NameNode

master@master:~/workspace/hadoop-2.7.1$

1. 把NameNode1的数据从master同步到lilei上

在lilei.hadoop-cluster上运行bin/hdfs namenode –bootstrapStandby，结果如下：

master@lilei:~/workspace/hadoop-2.7.1$ cd workspace/hadoop-2.7.1/

15/08/15 19:21:46 INFO namenode.NameNode: SHUTDOWN\_MSG:

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

SHUTDOWN\_MSG: Shutting down NameNode at lilei.hadoop-cluster/115.154.138.31

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

master@lilei:~/workspace/hadoop-2.7.1$

1. 停止master上的namenode，使用sbin/start-dfs.sh启动所有的namenode、datanode、journalnode和zkfc:

master@master:~/workspace/hadoop-2.7.1$ sbin/start-dfs.sh

Starting namenodes on [master.hadoop-cluster lilei.hadoop-cluster]

master.hadoop-cluster: starting namenode, logging to /home/master/workspace/hadoop-2.7.1/logs/hadoop-master-namenode-master.hadoop-cluster.out

lilei.hadoop-cluster: starting namenode, logging to /home/master/workspace/hadoop-2.7.1/logs/hadoop-master-namenode-lilei.out

hanmei.hadoop-cluster: starting datanode, logging to /home/master/workspace/hadoop-2.7.1/logs/hadoop-master-datanode-hanmei.out

lily.hadoop-cluster: starting datanode, logging to /home/master/workspace/hadoop-2.7.1/logs/hadoop-master-datanode-lily.out

lucy.hadoop-cluster: starting datanode, logging to /home/master/workspace/hadoop-2.7.1/logs/hadoop-master-datanode-lucy.out

Starting journal nodes [master.hadoop-cluster lilei.hadoop-cluster hanmei.hadoop-cluster]

lilei.hadoop-cluster: starting journalnode, logging to /home/master/workspace/hadoop-2.7.1/logs/hadoop-master-journalnode-lilei.out

master.hadoop-cluster: starting journalnode, logging to /home/master/workspace/hadoop-2.7.1/logs/hadoop-master-journalnode-master.hadoop-cluster.out

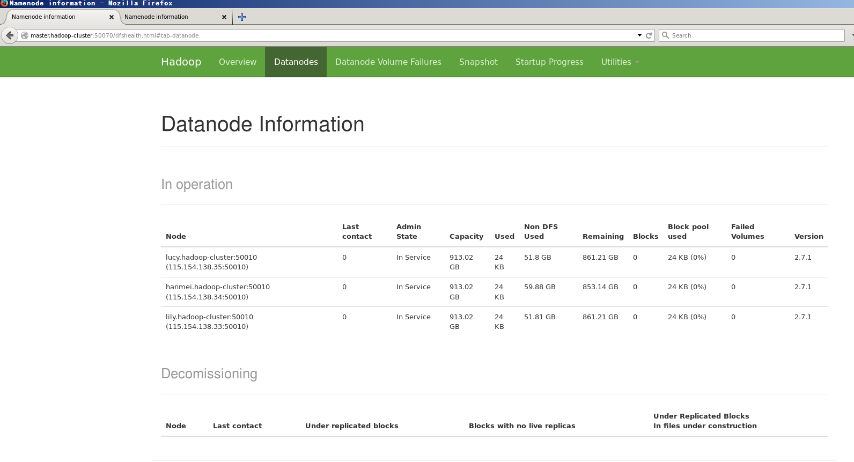
hanmei.hadoop-cluster: starting journalnode, logging to /home/master/workspace/hadoop-2.7.1/logs/hadoop-master-journalnode-hanmei.out

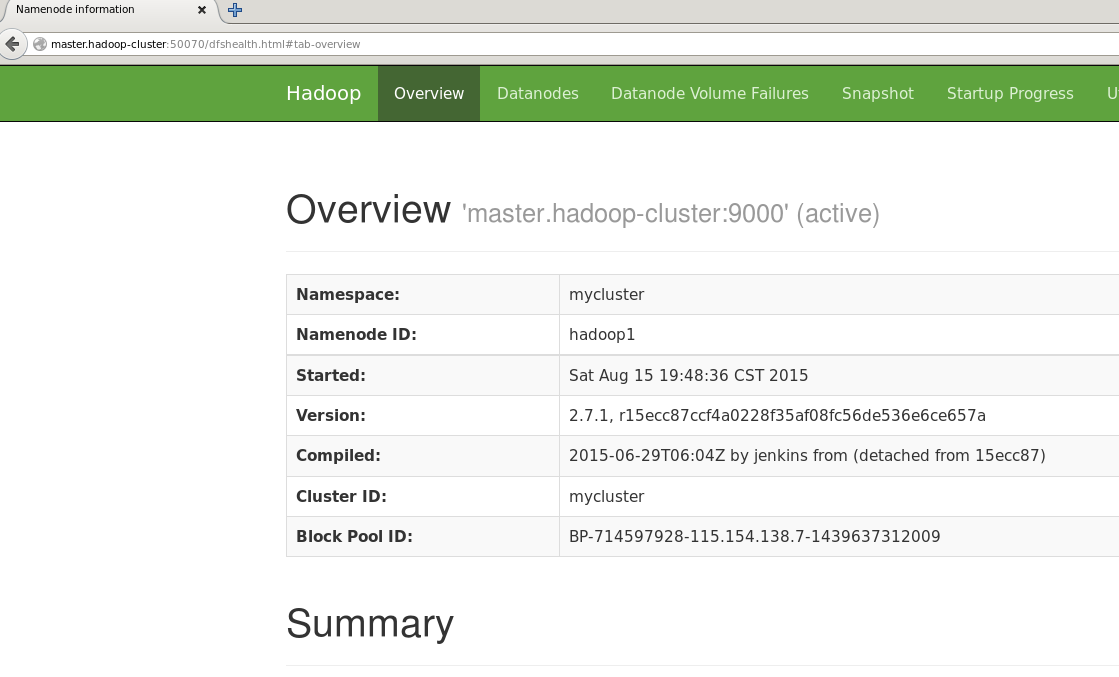
Starting ZK Failover Controllers on NN hosts [master.hadoop-cluster lilei.hadoop-cluster]

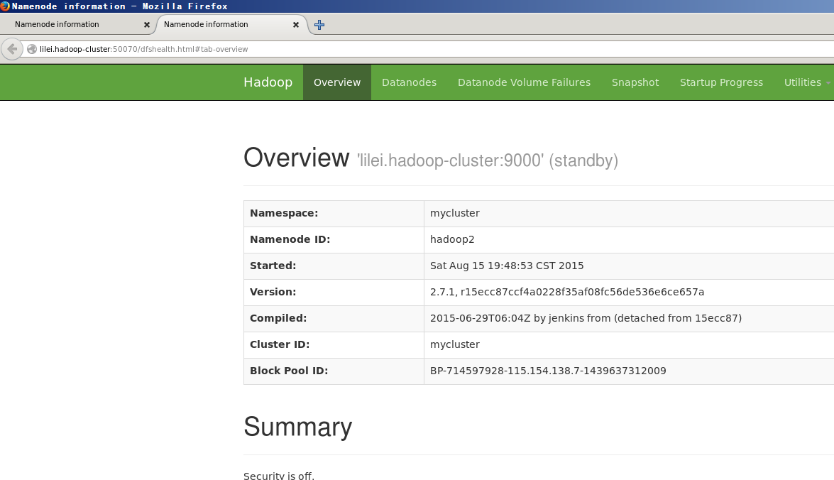
master.hadoop-cluster: starting zkfc, logging to /home/master/workspace/hadoop-2.7.1/logs/hadoop-master-zkfc-master.hadoop-cluster.out

lilei.hadoop-cluster: starting zkfc, logging to /home/master/workspace/hadoop-2.7.1/logs/hadoop-master-zkfc-lilei.out

master@master:~/workspace/hadoop-2.7.1$







从图中可以看出，两个namdenode都启动了，而且master处于active，而lilei处于standby，三个datanode都启动了。

1. 验证hdfs ha和自动failover

master@master:~/workspace/hadoop-2.7.1$ jps

18828 Jps

15662 NameNode

16227 DFSZKFailoverController

15209 QuorumPeerMain

15959 JournalNode

master@master:~/workspace/hadoop-2.7.1$ kill -9 15662

master@master:~/workspace/hadoop-2.7.1$ jps

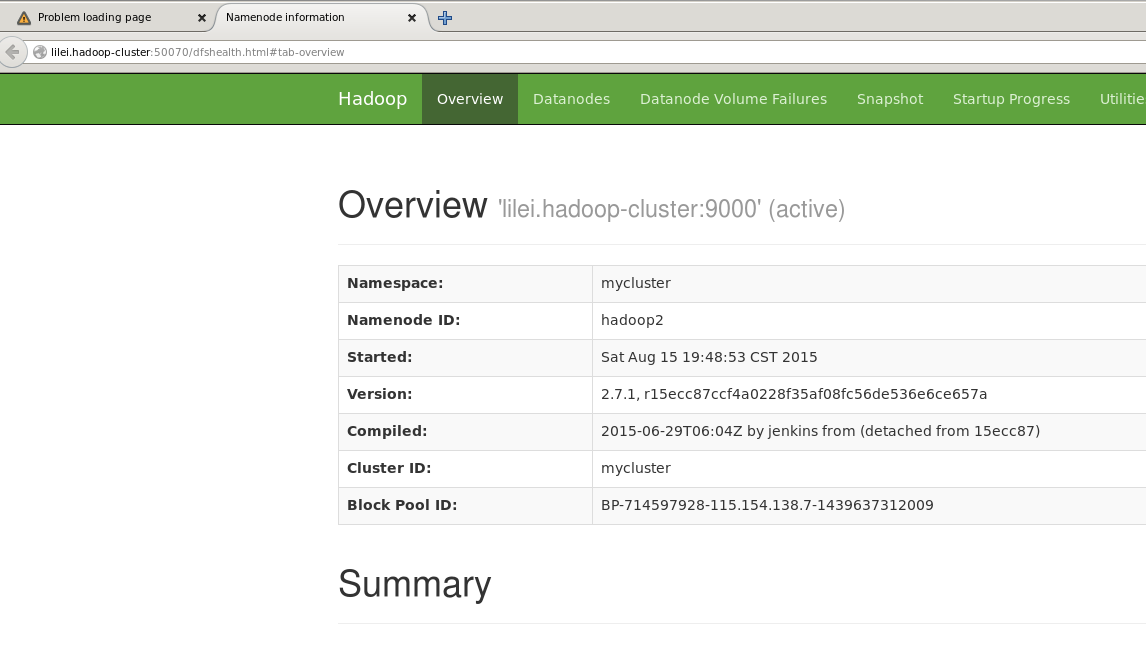
18873 Jps

16227 DFSZKFailoverController

15209 QuorumPeerMain

15959 JournalNode

master@master:~/workspace/hadoop-2.7.1$



可以看到lilei状态自动转为active。

重启master的namenode，发现其状态为standby:

master@master:~/workspace/hadoop-2.7.1$ sbin/hadoop-daemon.sh start namenode

starting namenode, logging to /home/master/workspace/hadoop-2.7.1/logs/hadoop-master-namenode-master.hadoop-cluster.out

master@master:~/workspace/hadoop-2.7.1$ jps

16227 DFSZKFailoverController

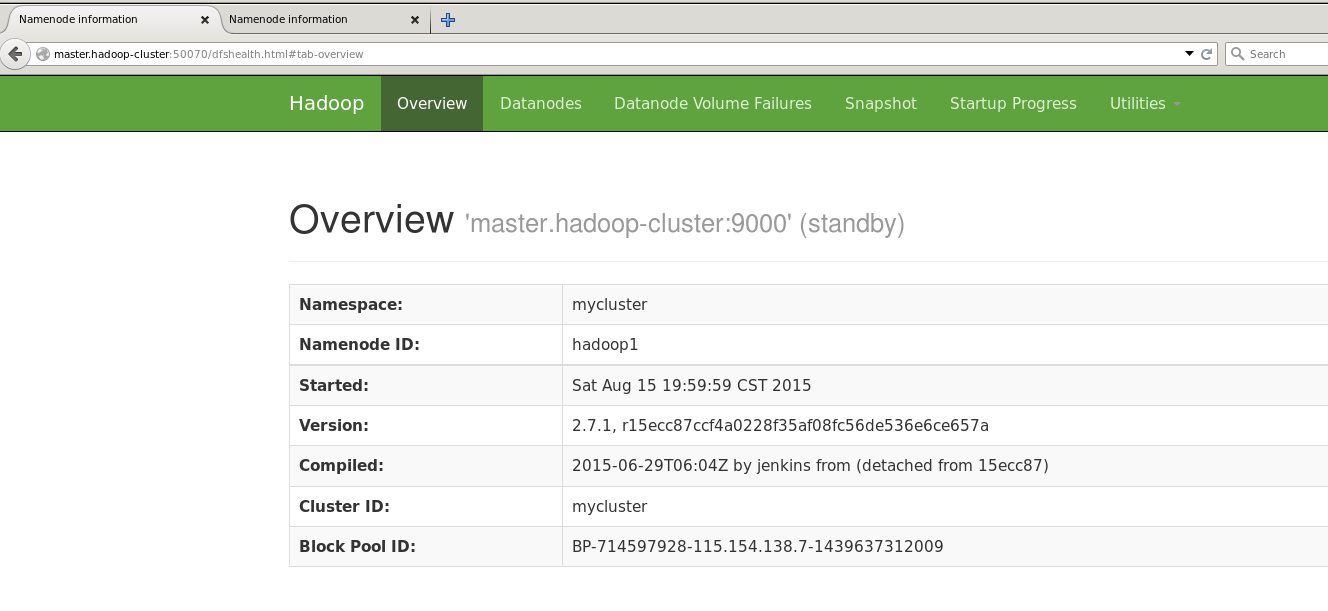
15209 QuorumPeerMain

15959 JournalNode

19715 NameNode

20081 Jps

master@master:~/workspace/hadoop-2.7.1$



1. 验证yarn的ha
2. 在master节点上运行sbin/start-yarn.sh，然后转到lilei节点上运行sbin/hadoop-daemon.sh start resourcemanager

master@master:~/workspace/hadoop-2.7.1$ sbin/start-yarn.sh

starting yarn daemons

starting resourcemanager, logging to /home/master/workspace/hadoop-2.7.1/logs/yarn-master-resourcemanager-master.hadoop-cluster.out

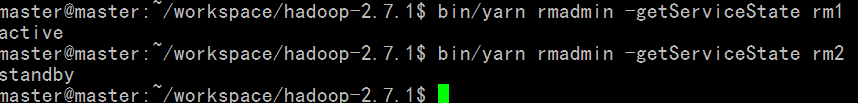
lily.hadoop-cluster: starting nodemanager, logging to /home/master/workspace/hadoop-2.7.1/logs/yarn-master-nodemanager-lily.out

hanmei.hadoop-cluster: starting nodemanager, logging to /home/master/workspace/hadoop-2.7.1/logs/yarn-master-nodemanager-hanmei.out

lucy.hadoop-cluster: starting nodemanager, logging to /home/master/workspace/hadoop-2.7.1/logs/yarn-master-nodemanager-lucy.out

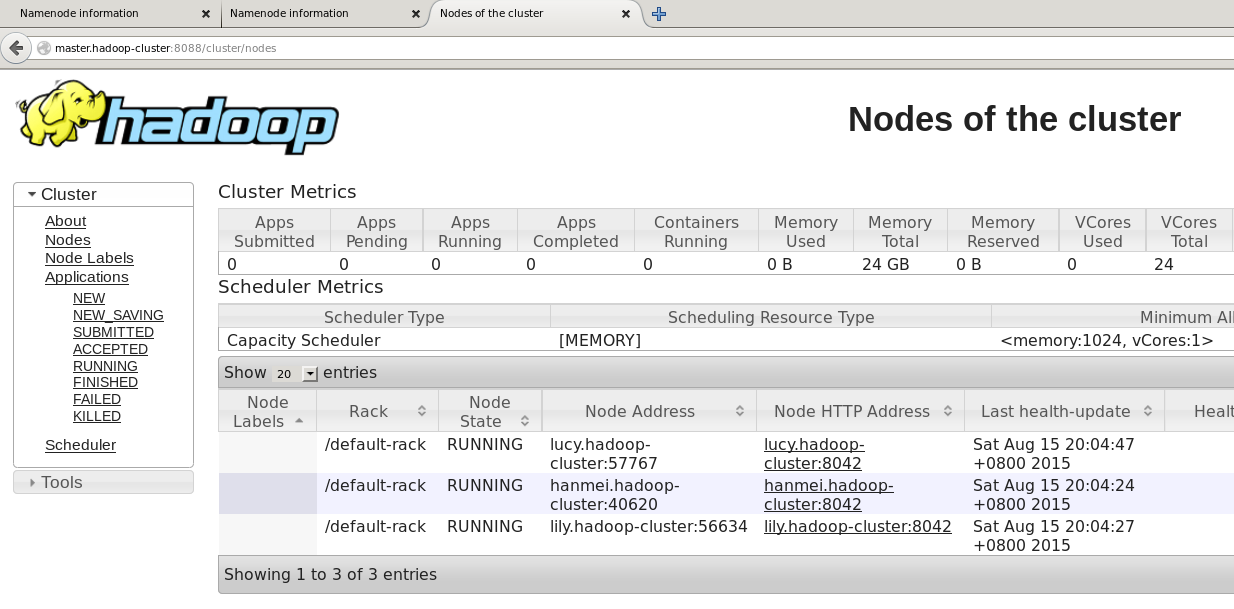
master@master:~/workspace/hadoop-2.7.1$

master@lilei:~/workspace/hadoop-2.7.1$ bin/yarn resourcemanager start



rm1为master.hadoop-cluster节点，rm2为lilei.hadoop-cluster节点。

启动lilei节点的resourcemanger以后访问lilei.hadoop-cluster:8088会自动跳转到master.hadoop-cluster:8088。



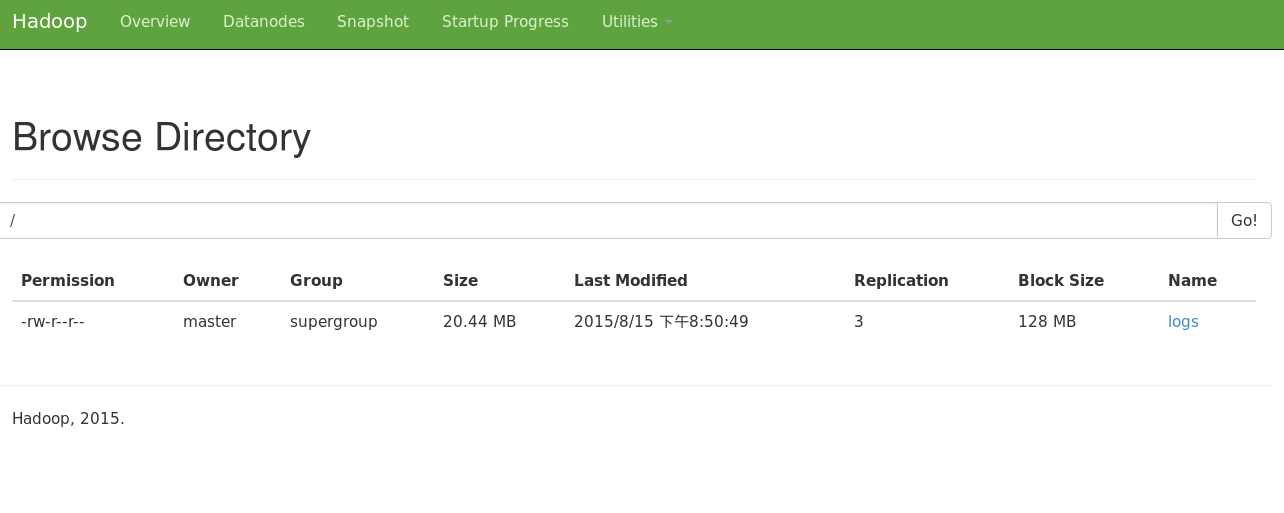
可以看到三个nodemanager也启动了。用和验证hdfs 自动failover一样的方法可以验证yarn的自动failover也是正常的。

1. 使用mapreduce自带的sample验证整体：
2. Wordcount程序

上传wordcount的输入文件：

master@master:~/workspace/hadoop-2.7.1$ bin/hdfs dfs -put logs/hadoop-master-namenode-master.hadoop-cluster.log /logs

上传完成可以在hdfs上看到logs文件：



运行程序：

master@master:~/workspace/hadoop-2.7.1$ bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-2.7.1.jar wordcount /logs /wcoutput

…

File Input Format Counters

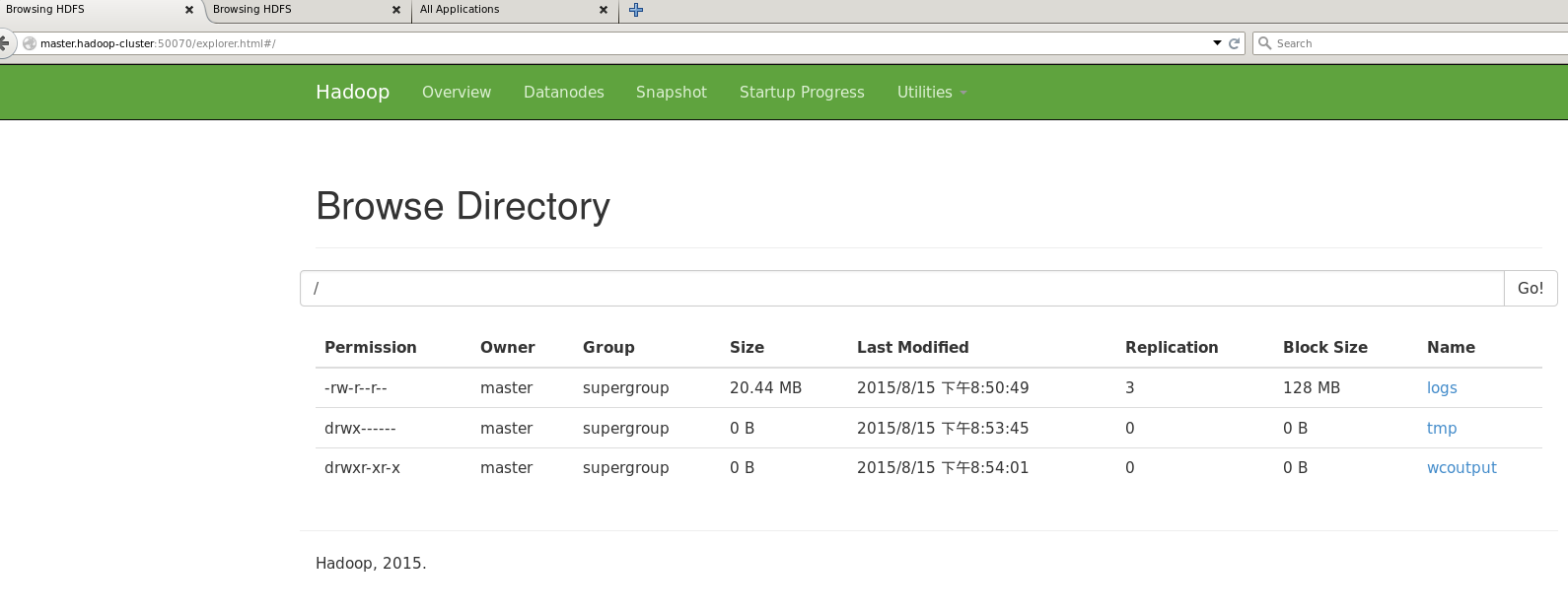
Bytes Read=21435578

File Output Format Counters

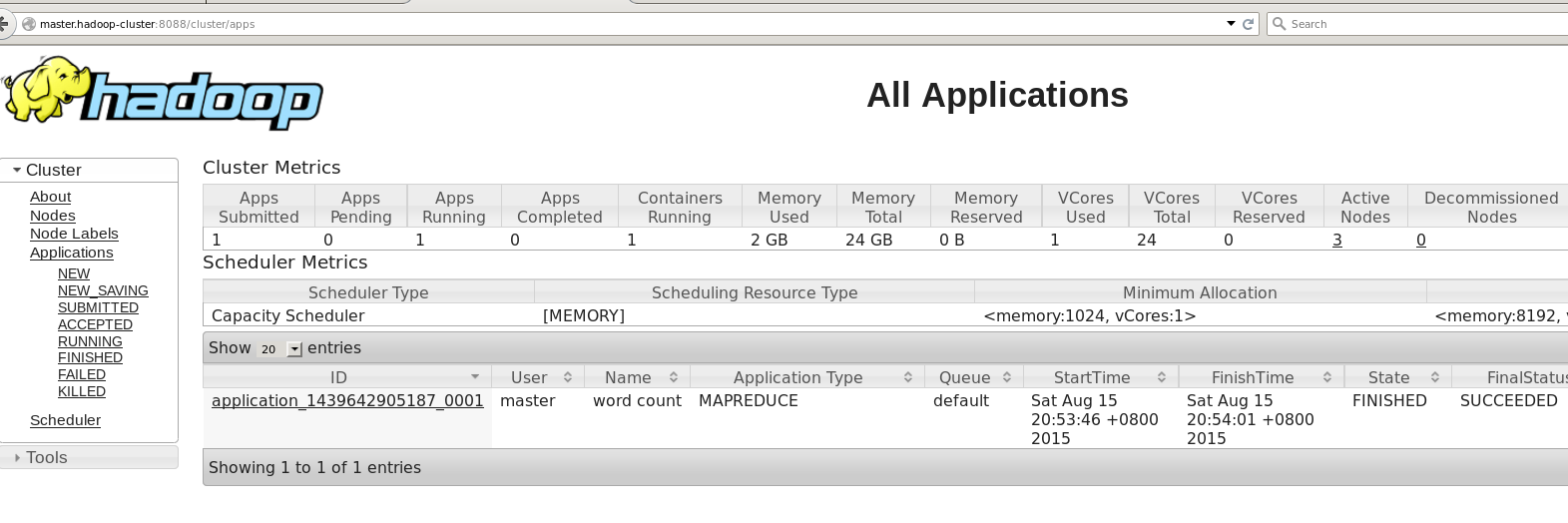
Bytes Written=1899757

master@master:~/workspace/hadoop-2.7.1$

结果运行正常，可以看到输出文件保存在hdfs上：



Yarn的web上也能看到程序的运行：

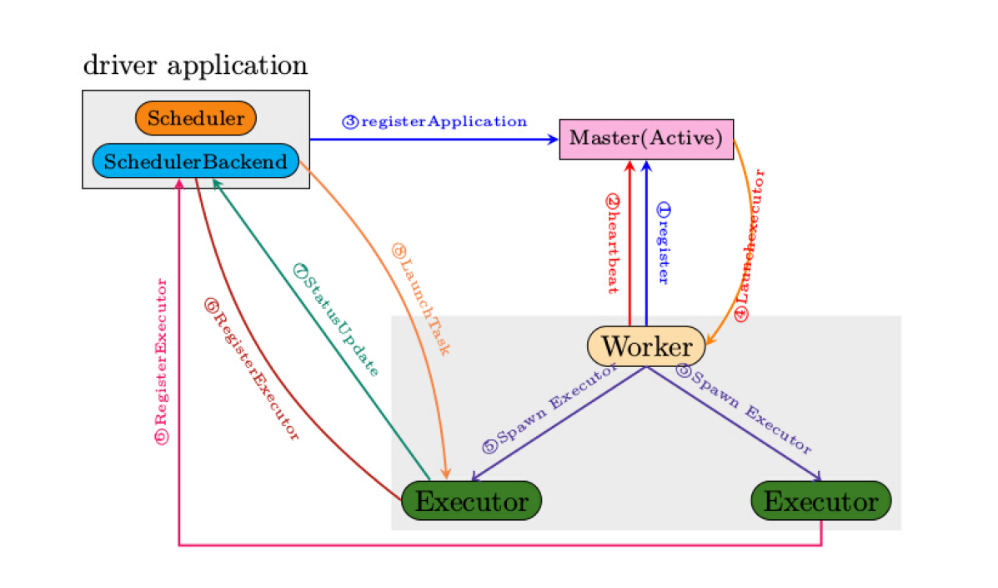


到此，所有的hadoop的hdfs和yarn和HA都实现了。

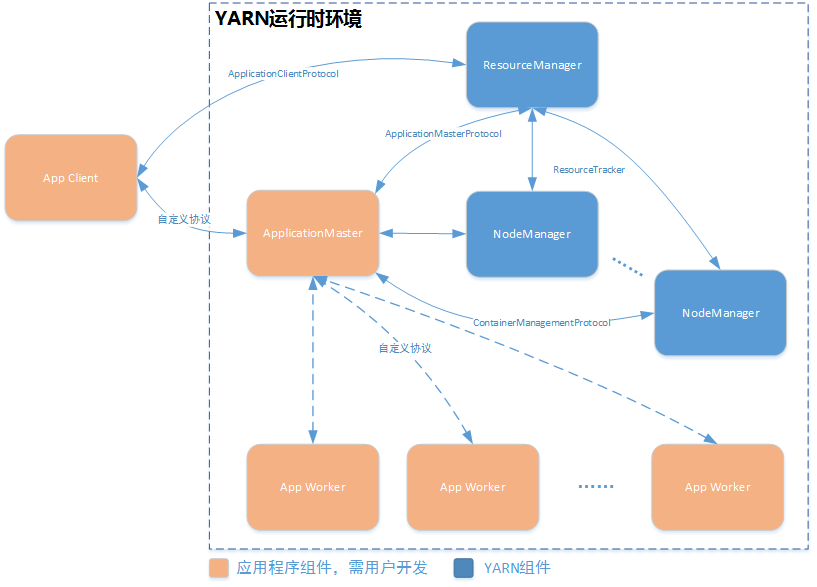
### Spark安装配置

1. 配置

需要修改两个文件，都位于spark目录下的conf文件夹下：slaves和spark-env.sh，其实这里这是在使用standalone模式是才会用到，为了测试spark的完整性，我们会运行一次standalone模式，但我们不会配置该模式的HA。因为最终我们的spark是运行在yarn上的，这也就是我们为什么不配置HA，相信从下面的两张图更能看得出来，第一张是spark的standalone模式的runtime图，第二张是spark on yarn的runtime图：



standalone runtime图



spark on yarn runtime图

scala-2.11.7在前面已经安装。

Slaves文件：

master@master:~/workspace/hadoop-2.7.1/app/spark-1.4.1-bin-hadoop2.6$ cat conf/slaves

# A Spark Worker will be started on each of the machines listed below.

lilei.hadoop-cluster

hanmei.hadoop-cluster

lucy.hadoop-cluster

lily.hadoop-cluster

spark-env.sh文件：

master@master:~/workspace/hadoop-2.7.1/app/spark-1.4.1-bin-hadoop2.6$ sed -n '/^[^#]/p' conf/spark-env.sh

export JAVA\_HOME=/usr/lib/jvm/java-1.7.0-openjdk-amd64

export SCALA\_HOME=/usr/bin

export HADOOP\_CONF\_DIR="/home/master/workspace/hadoop-2.7.1/etc/hadoop"

spark-env.sh主要配置的HADOOP\_CONF\_DIR环境变量，主要用来Hadoop集群的户主要配置文件位置，在与运行spark on yarn时会用到。将配置好的Spark复制到其他节点上的相应位置。

1. Standalone测试
2. 启动主节点上的master进程

master@master:~/workspace/hadoop-2.7.1/app/spark-1.4.1-bin-hadoop2.6$ sbin/start-master.sh

starting org.apache.spark.deploy.master.Master, logging to /home/master/workspace/hadoop-2.7.1/app/spark-1.4.1-bin-hadoop2.6/sbin/../logs/spark-master-org.apache.spark.deploy.master.Master-1-master.hadoop-cluster.out

master@master:~/workspace/hadoop-2.7.1/app/spark-1.4.1-bin-hadoop2.6$

master@master:~/workspace/hadoop-2.7.1/app/spark-1.4.1-bin-hadoop2.6$ jps

9772 NameNode

2614 QuorumPeerMain

10342 DFSZKFailoverController

20358 Master

10078 JournalNode

10793 ResourceManager

20744 Jps

master@master:~/workspace/hadoop-2.7.1/app/spark-1.4.1-bin-hadoop2.6$

1. 启动所有从节点上的worker进程

master@master:~/workspace/hadoop-2.7.1/app/spark-1.4.1-bin-hadoop2.6$ sbin/start-slaves.sh

lilei.hadoop-cluster: starting org.apache.spark.deploy.worker.Worker, logging to /home/master/workspace/hadoop-2.7.1/app/spark-1.4.1-bin-hadoop2.6/sbin/../logs/spark-master-org.apache.spark.deploy.worker.Worker-1-lilei.out

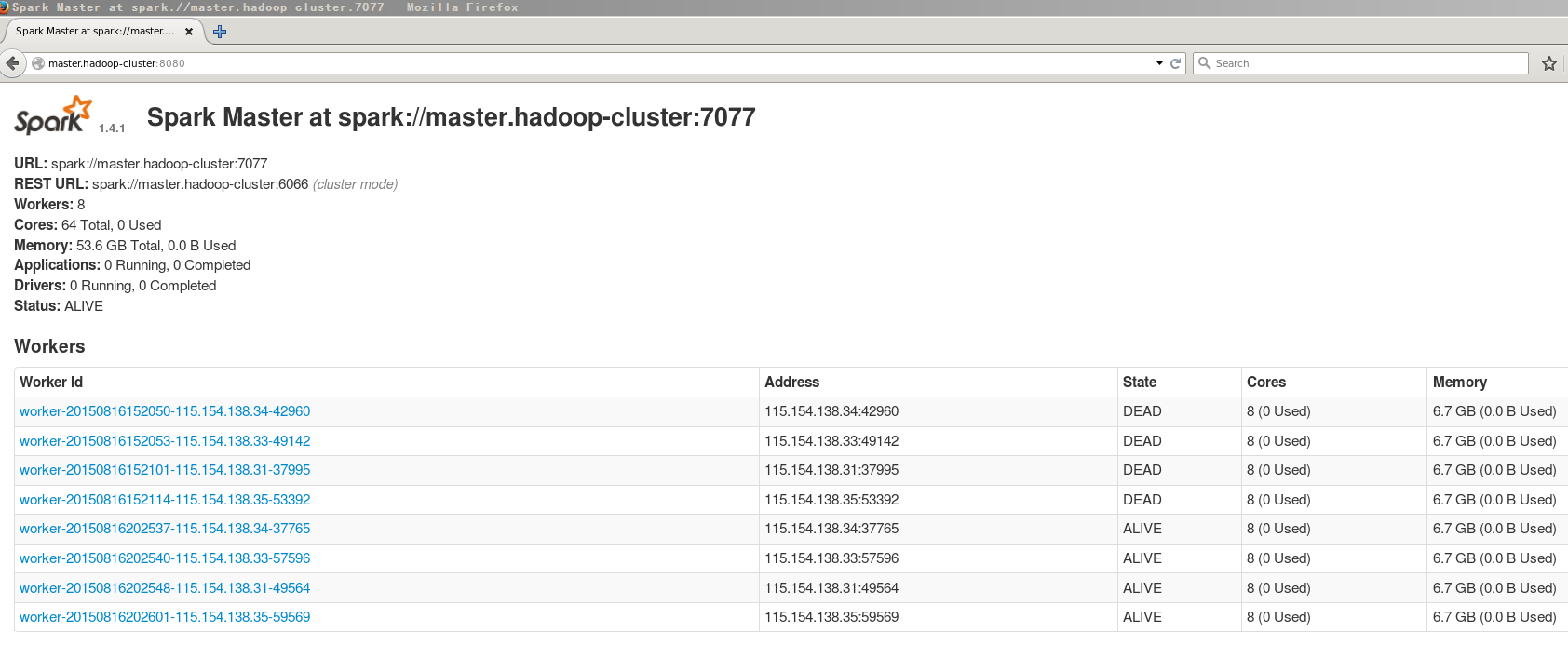
lucy.hadoop-cluster: starting org.apache.spark.deploy.worker.Worker, logging to /home/master/workspace/hadoop-2.7.1/app/spark-1.4.1-bin-hadoop2.6/sbin/../logs/spark-master-org.apache.spark.deploy.worker.Worker-1-lucy.out

hanmei.hadoop-cluster: starting org.apache.spark.deploy.worker.Worker, logging to /home/master/workspace/hadoop-2.7.1/app/spark-1.4.1-bin-hadoop2.6/sbin/../logs/spark-master-org.apache.spark.deploy.worker.Worker-1-hanmei.out

lily.hadoop-cluster: starting org.apache.spark.deploy.worker.Worker, logging to /home/master/workspace/hadoop-2.7.1/app/spark-1.4.1-bin-hadoop2.6/sbin/../logs/spark-master-org.apache.spark.deploy.worker.Worker-1-lily.out

master@master:~/workspace/hadoop-2.7.1/app/spark-1.4.1-bin-hadoop2.6$

都启动成功以后就能在master的web UI上看到所有的worker(上面四个是我初次测试时启动的节点，可以看到他们的State是DEAD的)：

1. 启动spark-shell

master@master:~/workspace/hadoop-2.7.1/app/spark-1.4.1-bin-hadoop2.6$ bin/spark-shell

15/08/16 20:35:41 INFO metastore.ObjectStore: Initialized ObjectStore

15/08/16 20:35:42 WARN metastore.ObjectStore: Version information not found in metastore. hive.metastore.schema.verification is not enabled so recording the schema version 0.13.1aa

15/08/16 20:35:43 INFO metastore.HiveMetaStore: Added admin role in metastore

15/08/16 20:35:43 INFO metastore.HiveMetaStore: Added public role in metastore

15/08/16 20:35:43 INFO metastore.HiveMetaStore: No user is added in admin role, since config is empty

15/08/16 20:35:43 INFO session.SessionState: No Tez session required at this point. hive.execution.engine=mr.

15/08/16 20:35:43 INFO repl.SparkILoop: Created sql context (with Hive support)..

SQL context available as sqlContext.

scala>

运行一个小程序(程序中文件从hdfs中读取)：

scala> val textFile = sc.textFile("hdfs://master.hadoop-cluster/logs") //加载数据文件，从HDFS路径读取

15/08/16 20:38:58 INFO storage.MemoryStore: ensureFreeSpace(90688) called with curMem=0, maxMem=278302556

15/08/16 20:38:58 INFO storage.MemoryStore: Block broadcast\_0 stored as values in memory (estimated size 88.6 KB, free 265.3 MB)

15/08/16 20:38:58 INFO storage.MemoryStore: ensureFreeSpace(20759) called with curMem=90688, maxMem=278302556

15/08/16 20:38:58 INFO storage.MemoryStore: Block broadcast\_0\_piece0 stored as bytes in memory (estimated size 20.3 KB, free 265.3 MB)

15/08/16 20:38:58 INFO storage.BlockManagerInfo: Added broadcast\_0\_piece0 in memory on localhost:36506 (size: 20.3 KB, free: 265.4 MB)

15/08/16 20:38:58 INFO spark.SparkContext: Created broadcast 0 from textFile at <console>:21

textFile: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[1] at textFile at <console>:21

scala> textFile.count() //列出文件行数

15/08/16 20:39:36 INFO scheduler.TaskSetManager: Finished task 0.0 in stage 0.0 (TID 0) in 272 ms on localhost (2/2)

15/08/16 20:39:36 INFO scheduler.DAGScheduler: ResultStage 0 (count at <console>:24) finished in 0.276 s

15/08/16 20:39:36 INFO scheduler.TaskSchedulerImpl: Removed TaskSet 0.0, whose tasks have all completed, from pool

15/08/16 20:39:36 INFO scheduler.DAGScheduler: Job 0 finished: count at <console>:24, took 0.316608 s

res0: Long = 89668 //一共89668行

scala>

1. 测试spark on yarn模式(求pi的值)：

master@master:~/workspace/hadoop-2.7.1/app/spark-1.4.1-bin-hadoop2.6$ ./bin/spark-submit --class org.apache.spark.examples.SparkPi \

--master yarn-cluster \

--num-executors 3 \

--driver-memory 4g \

--executor-memory 2g \

--executor-cores 1 \

--queue thequeue \

lib/spark-examples\*.jar

10000

……

15/08/16 20:45:52 INFO yarn.Client: Application report for application\_1439711688444\_0012 (state: RUNNING)

15/08/16 20:45:53 INFO yarn.Client: Application report for application\_1439711688444\_0012 (state: FINISHED)

15/08/16 20:45:53 INFO yarn.Client:

client token: N/A

diagnostics: N/A

ApplicationMaster host: 115.154.138.33

ApplicationMaster RPC port: 0

queue: default

start time: 1439729116204

final status: SUCCEEDED

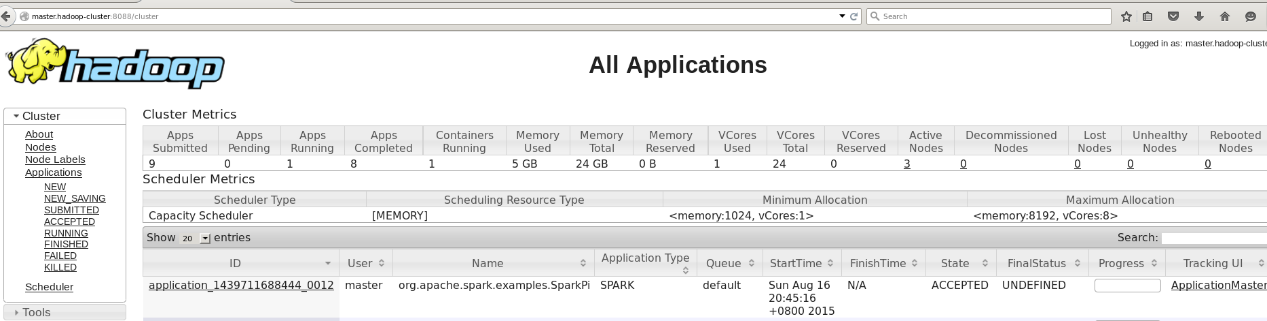
tracking URL: http://master.hadoop-cluster:8088/proxy/application\_1439711688444\_0012/

user: master

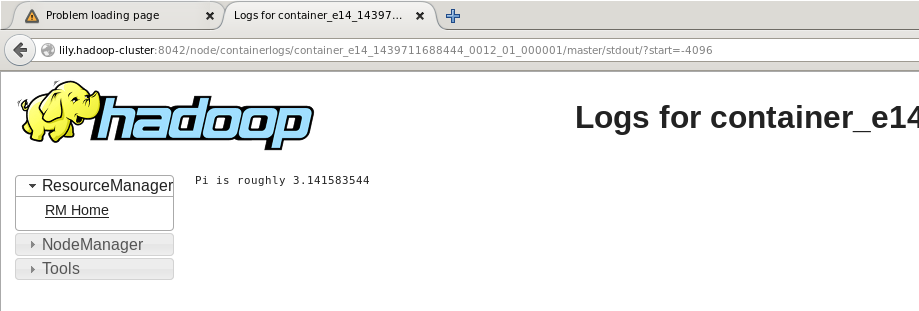
15/08/16 20:45:53 INFO util.Utils: Shutdown hook called

15/08/16 20:45:53 INFO util.Utils: Deleting directory /tmp/spark-8380c531-7e00-478f-b0bf-178e17e10c88

master@master:~/workspace/hadoop-2.7.1/app/spark-1.4.1-bin-hadoop2.6$

可以在yarn上看到提交的作业：

运行结果:



到此，所有的工作就完成了。行文匆忙，如有错误之处，欢迎交流：[wearyoung@outlook.com](mailto:wearyoung@outlook.com)。