

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 daemon.)，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。

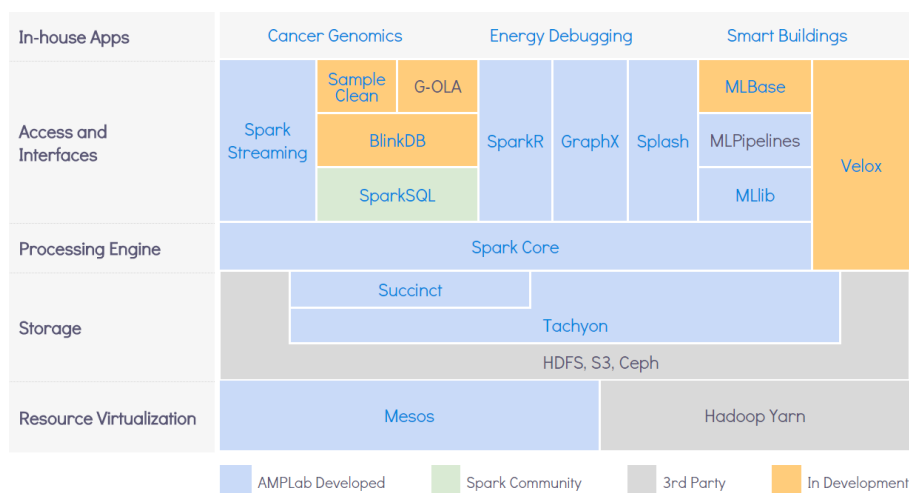


图 1-BDAS

下面就是目前要搭建的平台框架：

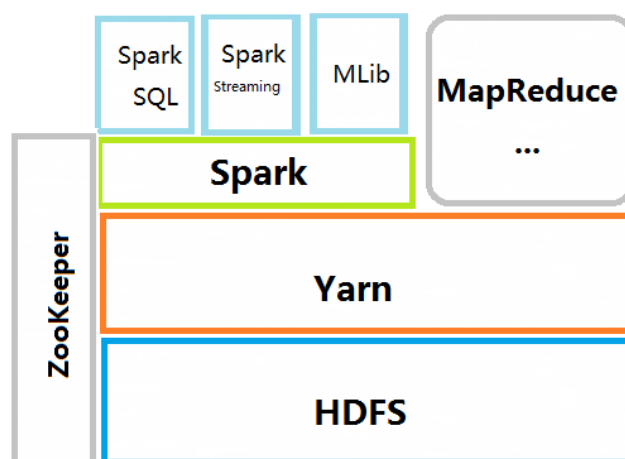


图 2-平台设计框架

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. 给每台机器创建相同的用户名，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。

2. 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 authorized_keys 文件中，这里要用到四个脚本：noscpc.exp、writeauthkey.exp、remotesudo.exp 和 createSshNopasswd.sh。

代码如下：

noscpc.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 noscpc.exp
#!/usr/bin/expect
#./noscpc.exp localfile remotefile
#scp file without password(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'`
    do

```

```

                if grep $var $file 1>/dev/null 2>/dev/null
                then
                    ./noscpc.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

```

3. 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=5adoop-cluster
pwd=123456
file=/etc/hosts
localfile=~/.ssh/id_rsa.pub
username=master
if [ -e "$localfile" ]
then
    for var in `grep "5 adoop" ${file} | grep -v "master" | sed 's/^.* 5 adoop-cluster //'`
2>/dev/null`
    do
        ./remotesudo.exp "sed -i '\${a}\${masterip} master.${endfix} master' /etc/hosts"
        ${var}.${endfix} ${username} ${pwd}
        IFS.OLD=$IFS
        IFS=$'\n'
        for iphost in `grep "5adoop" ${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
done
fi

```

4. 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 //'`
2>/dev/null`
    do
        if grep $var $file 1>/dev/null 2>/dev/null
        then
            ${commdir}/noscpc.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
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 //' 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
```

5. 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 //' 2>/dev/null`
    do
        ./remotesudoNoWait.exp "apt-get -y install openjdk-7-jre" ${var}.${endfix}
    done
    ./remotesudoNoWait.exp "apt-get -y install expect" ${var}.${endfix} ${username}
    ./remotesudoNoWait.exp "ssh ${var}.${endfix} grep 'JAVA_HOME' /etc/profile"
    status=$?
    if [ $status -ne 0 ]
    then
        ./remotesudoNoWait.exp "ssh ${var}.${endfix} sed -i '\$a\\export JAVA_HOME=$java_home'
        /etc/profile" ${var}.${endfix} ${username} ${pwd}
    fi

    ./remotesudoNoWait.exp "ssh ${var}.${endfix} which scala"
    if [ $? -ne 0 ]
    then
        ./remotesudoNoWait.exp "scp -r $scala master@${var}.${endfix}:${scala_dir}"
        ./remotesudoNoWait.exp "ssh master@${var}.${endfix} tar -xzf $scala -C $scala_dir/"
        ./remotesudoNoWait.exp "ln -s $scala_dir/scala-2.11.7/bin/scala /usr/bin/"
    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
noscpc.exp      remotesudo.exp
clearTmp.sh     remotesudoNoWait.exp
cpFilesToSlaves.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 7adoop-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

- 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/zkdata-log
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；server1、server2 和 server3 即为 ZooKeeper 集群的所有节点。

- 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，按照 zoo.cfg 的配置，依此写入相应的数字。zkdata-log 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。

- hadoop-env.sh 文件

添加 JDK 环境变量

```
export JAVA_HOME=/usr/lib/jvm/java-1.7.0-openjdk-amd64
```

- 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>

```

c. 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>
    <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>

```

d. 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>

```

e. 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 YARN 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>

```

f. slave 文件

```

hanmei.hadoop-cluster
lucy.hadoop-cluster
lily.hadoop-cluster

```

3. 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 //'`
    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

```

4. 登录到 ZooKeeper 的其他节点，修改前面我们配置 ZooKeeper 时说到的 myid 文件。

5. 测试

a. 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。

b. 验证格式化 zookeeper

1. 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$SendThread@975] - Opening socket connection to server ip6-
localhost/0:0:0:0:0:0:1:2181. Will not attempt to authenticate using SASL (unknown error)
JLine support is enabled
2015-08-15 17:31:09,071 [myid:] - INFO [main-SendThread(ip6-
localhost:2181):ClientCnxn$SendThread@852] - Socket connection established to ip6-
localhost/0:0:0:0:0:0:1:2181, 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/0:0:0:0:0:0:1:2181, 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。

2. 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]
```

c. 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$
```

d. 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$ jps
31852 QuorumPeerMain
4877 JournalNode
31223 Jps
31102 NameNode
master@master:~/workspace/hadoop-2.7.1$
```

e. 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$
```

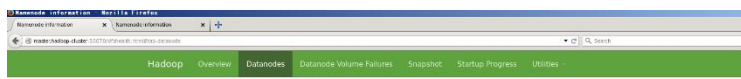
f. 把 NameNode1 的数据从 master 同步到 lilei 上

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$
```

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

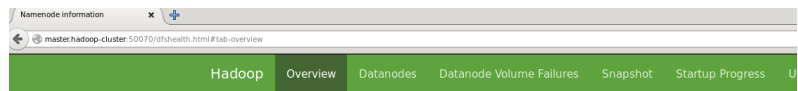
```
master@master:~/workspace/hadoop-2.7.1$ sbin/start-dfs.sh
Starting namenodes on [master.hadoop-cluster lilei.hadoop-cluster]
master.hadoop-cluster: starting namenode, logging to /home/master/workspace/hadoop-2.7.1/logs/hadoop-master-namenode-master.hadoop-cluster.out
lilei.hadoop-cluster: starting 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$
```



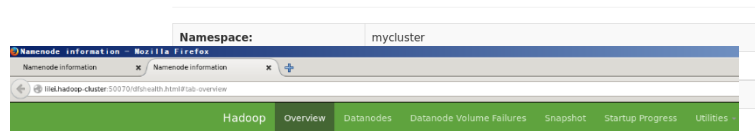
The screenshot shows the Hadoop web interface with the 'Datanode Information' tab selected. It displays a table of datanodes in operation and a section for decommissioning.

In operation									
Node	Last contact	Admin State	Capacity	Used	Rep. DPS	Remaining Blocks	Block pool used	Failed Volumes	Version
lucy.hadoop-cluster/20010 (115.154.138.35-50030)	0	In Service	512.02 GB	24 KB	51.9 GB	893.22 GB	0	24 KB (9%)	0
hanmei.hadoop-cluster/50010 (115.154.138.34-50030)	0	In Service	512.02 GB	24 KB	50.88 GB	853.14 GB	0	24 KB (9%)	0
lily.hadoop-cluster/50010 (115.154.138.33-50030)	0	In Service	512.02 GB	24 KB	51.83 GB	883.21 GB	0	24 KB (9%)	0

Node	Last contact	Under replicated blocks	Blocks with no live replicas	Under Replicated Blocks in flux under construction
------	--------------	-------------------------	------------------------------	--



Overview 'master.hadoop-cluster:9000' (active)



Overview 'lilei.hadoop-cluster:9000' (standby)

Namespace:	mycluster
Namespace ID:	hadoop2
Started:	Sat Aug 15 19:48:53 CST 2015
Version:	2.7.1, r15ecc87ccf4a0228f35af08fc56de536e6ce657a
Compiled:	2015-06-29T06:04Z by jenkins from (detached from 15ecc87)
Cluster ID:	mycluster
Block Pool ID:	BP-714597928-115.154.138.7-1439637312009

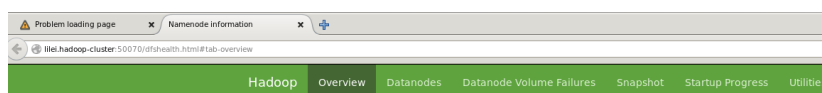
Summary

Security is off.

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

h. 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$
```



Overview 'lilei.hadoop-cluster:9000' (active)

Namespace:	mycluster
Namespace ID:	hadoop2
Started:	Sat Aug 15 19:48:53 CST 2015
Version:	2.7.1, r15ecc87ccf4a0228f35af08fc56de536e6ce657a
Compiled:	2015-06-29T06:04Z by jenkins from (detached from 15ecc87)
Cluster ID:	mycluster
Block Pool ID:	BP-714597928-115.154.138.7-1439637312009

Summary

可以看到 lilei 状态自动转为 active。

重启 master 的 namenode，发现其状态为 standby:

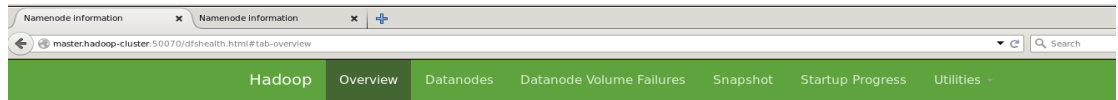
```
master@master:~/workspace/hadoop-2.7.1$ sbin/hadoop-daemon.sh start namenode
starting namenode, logging to /home/master/workspace/hadoop-2.7.1/logs/hadoop-master-
namenode-master.hadoop-cluster.out
```



```

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$

```



Overview 'master.hadoop-cluster:9000' (standby)

Namespace:	mycluster
Namenode ID:	hadoop1
Started:	Sat Aug 15 19:59:59 CST 2015
Version:	2.7.1, r15ecc87ccf4a0228f35af08fc56de536e6ce657a
Compiled:	2015-06-29T06:04Z by jenkins from (detached from 15ecc87)
Cluster ID:	mycluster
Block Pool ID:	BP-714597928-115.154.138.7-1439637312009

i. 验证 yarn 的 ha

1. 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

```

```

master@master:~/workspace/hadoop-2.7.1$ bin/yarn radmin -getServiceState rm1
active
master@master:~/workspace/hadoop-2.7.1$ bin/yarn radmin -getServiceState rm2
standby
master@master:~/workspace/hadoop-2.7.1$

```

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

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

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCoers Used	VCoers Total
0	0	0	0	0	0 B	24 GB	0 B	0	24

Node Labels	Rack	Node State	Node Address	Node HTTP Address	Last health-update	Heal
/default-rack		RUNNING	lily.hadoop-cluster:57767	lily.hadoop-cluster:8042	Sat Aug 15 20:04:47 +0800 2015	
/default-rack		RUNNING	hanmei.hadoop-cluster:40620	hanmei.hadoop-cluster:8042	Sat Aug 15 20:04:24 +0800 2015	
/default-rack		RUNNING	lily.hadoop-cluster:56634	lily.hadoop-cluster:8042	Sat Aug 15 20:04:27 +0800 2015	

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。

6. 使用 **mapreduce** 自带的 **sample** 验证整体:

a. **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:



Browse Directory

/							Go!
Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	master	supergroup	20.44 MB	2015/8/15 下午8:50:49	3	128 MB	logs
Hadoop, 2015.							

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

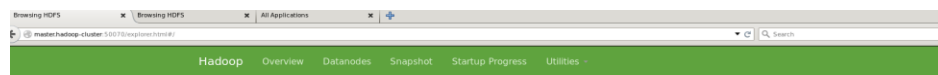
Bytes Read=21435578

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:



Browse Directory

/								Go
Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name	
-rw-r--r--	master	supergroup	20.44 MB	2015/8/15 下午8:50:49	3	128 MB	logs	
drwx-----	master	supergroup	0 B	2015/8/15 下午8:53:45	0	0 B	tmp	
drwxr-xr-x	master	supergroup	0 B	2015/8/15 下午8:54:01	0	0 B	wcoutput	

Hadoop, 2015.

Yarn 的 **web** 上也能看到程序的运行:

All Applications

Cluster Metrics		Cluster Metrics										
Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved	Active Nodes	Decommissioned Nodes
1	0	1	0	1	2 GB	24 GB	0 B	1	24	0	3	0

Scheduler Metrics

Scheduler Type: Capacity SchedulerScheduling Resource Type: [MEMORY]Minimum Allocation: <memory:1024, vCores:1><memory:8192, vCores:1>

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus
application_1439642905187_0001	master	word count	MAPREDUCE	default	Sat Aug 15 20:53:46 +0800 2015	Sat Aug 15 20:54:01 +0800 2015	FINISHED	SUCCEEDED

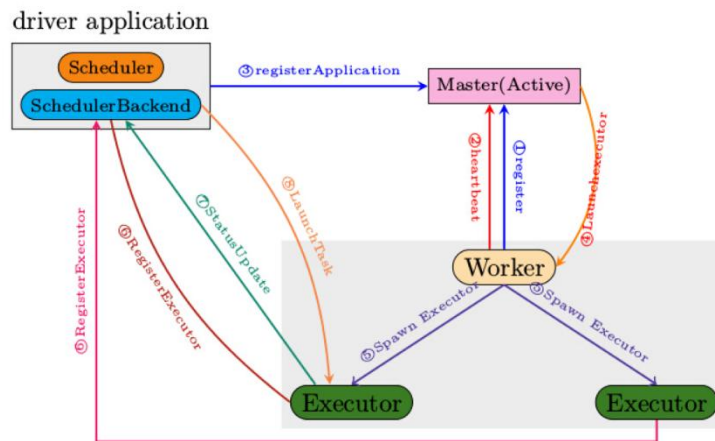
Showing 1 to 1 of 1 entries

到此, 所有的 **hadoop** 的 **hdfs** 和 **yarn** 和 **HA** 都实现了。

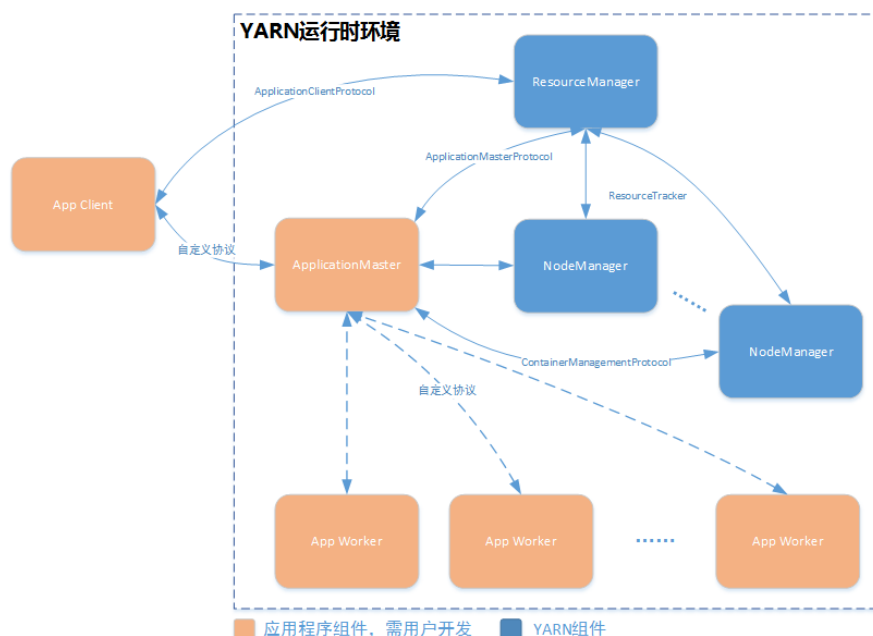
Spark 安装配置

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。

2. Standalone 测试

a. 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$

```

b. 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):

Worker Id	Address	State	Cores	Memory
worker-20150816152050-115.154.138.34-42960	115.154.138.34:42960	DEAD	8 (0 Used)	6.7 GB (0.0 B Used)
worker-20150816152053-115.154.138.33-49142	115.154.138.33:49142	DEAD	8 (0 Used)	6.7 GB (0.0 B Used)
worker-20150816152101-115.154.138.31-37995	115.154.138.31:37995	DEAD	8 (0 Used)	6.7 GB (0.0 B Used)
worker-20150816152114-115.154.138.35-53392	115.154.138.35:53392	DEAD	8 (0 Used)	6.7 GB (0.0 B Used)
worker-20150816202537-115.154.138.34-37765	115.154.138.34:37765	ALIVE	8 (0 Used)	6.7 GB (0.0 B Used)
worker-20150816202540-115.154.138.33-57596	115.154.138.33:57596	ALIVE	8 (0 Used)	6.7 GB (0.0 B Used)
worker-20150816202548-115.154.138.31-49564	115.154.138.31:49564	ALIVE	8 (0 Used)	6.7 GB (0.0 B Used)
worker-20150816202601-115.154.138.35-59569	115.154.138.35:59569	ALIVE	8 (0 Used)	6.7 GB (0.0 B Used)

c. 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.version 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>

```

3. 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 \

```

```

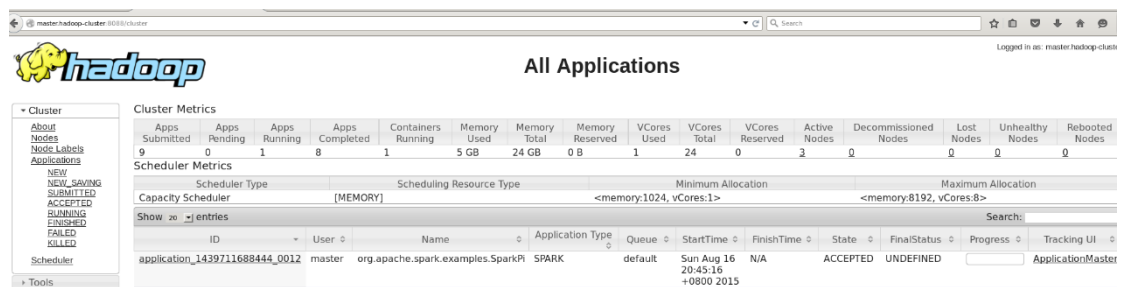
--executor-cores 1 \
--queue thequeue \
lib/spark-examples*.jar
10000

.....

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$

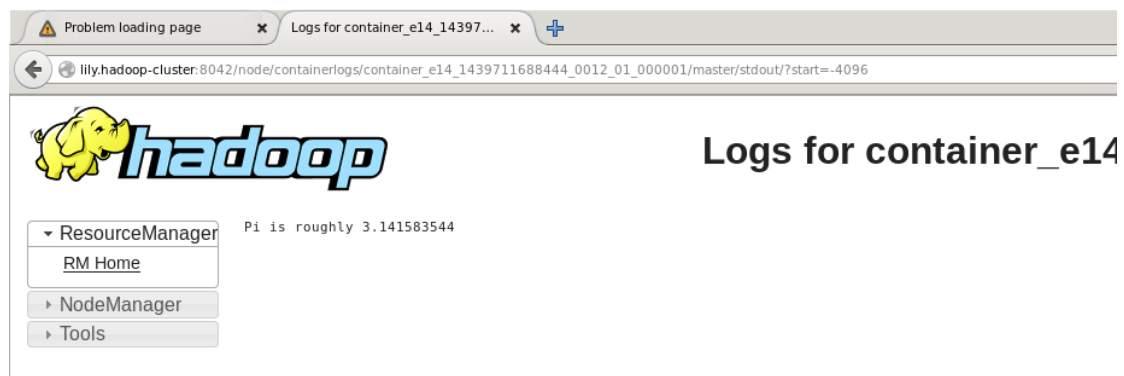
```

可以在 yarn 上看到提交的作业：



The screenshot shows the Hadoop YARN web interface. The top navigation bar includes the Hadoop logo and the title 'All Applications'. On the left, there is a sidebar with a 'Cluster' section containing links like 'About', 'Nodes', 'Node Labels', and 'Applications'. The main content area displays 'Cluster Metrics' and 'Scheduler Metrics'. Below these, there is a table of applications. The table has columns for ID, User, Name, Application Type, Queue, Start Time, Finish Time, State, Final Status, Progress, and Tracking UI. One application is listed: application_1439711688444_0012, submitted by master, named org.apache.spark.examples.SparkPi, of type SPARK, in the default queue, starting on Sun Aug 16 20:45:16 +0800 2015, with a state of ACCEPTED and final status of UNDEFINED.

运行结果：



The screenshot shows the Hadoop YARN web interface displaying the logs for a specific container. The top navigation bar includes the Hadoop logo and the title 'Logs for container_e14'. The main content area shows the logs for the container_e14_1439711688444_0012_01_000001. The logs show the output of the SparkPi application, which is 'Pi is roughly 3.141583544'. On the left, there is a sidebar with a 'ResourceManager' section containing links like 'RM Home', 'NodeManager', and 'Tools'.

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。