Hadoop-2.7.2 分布式安装手册

一见 2015/10/20

目录

1. 前言

当前版本的 Hadoop 已解决了 hdfs、yarn 和 hbase 等单点，并支持自动的主备切换。

本文的目的是为当前最新版本的 Hadoop 2.7.2 提供最为详细的安装说明，以帮助减少安装过程中遇到的困难，并对一些错误原因进行说明，hdfs 配置使用基于 QJM（Quorum Journal Manager）的 HA。本文的安装只涉及了 hadoop-common、hadoop-hdfs、hadoop-mapreduce 和 hadoop-yarn，并不包含 HBase、Hive 和 Pig 等。

2. 特性介绍

版本	发版本日期	新特性
2.7.2	2016/1/25	
2.7.1	2015/7/6	
2.7.0	2015/4/21	1） 不再支持 JDK6，须 JDK 7+ 2） 支持文件**截取**（truncate） 3） 支持为每种存储类型设置配额 4） 支持文件**变长块**（之前一直为固定块大小，默认为 64M） 5） 支持 Windows Azure Storage 6） YARN 认证可插拔 7） 自动共享，全局缓存 YARN 本地化资源（测试阶段） 8） 限制一个作业运行的 Map/Reduce 任务 9） 加快大量输出文件时大型作业的 FileOutputCommitter 速度
2.6.4	2016/2/11	
2.6.3	2015/12/17	

2.6.2	2015/10/28	
2.6.1	2015/9/23	
2.6.0	**2014/11/18**	**1) YARN 支持长时间运行的服务** **2) YARN 支持升级回滚** **3) YARN 支持应用运行在 Docker 容器中**
2.5.2	2014/11/19	
2.5.1	2014/9/12	
2.5.0	2014/8/11	
2.4.1	2014/6/30	
2.4.0	2014/4/7	1) HDFS 升级回滚 2) HDFS 支持完整的 https 3) YARN ResourceManager 支持自动故障切换
2.2.0	2013/10/15	1) HDFS Federation 2) HDFS Snapshots
2.1.0-beta	2013/8/25	1) HDFS 快照 2) 支持 Windows
2.0.3-alpha	**2013/2/14**	**1) 基于 QJM 的 NameNode HA**
2.0.0-alpha	**2012/5/23**	**1) 人工切换的 NameNode HA** **2) HDFS Federation**
1.0.0	**2011/12/27**	
0.23.11	2014/6/27	
0.23.10	2013/12/11	
0.22.0	2011/12/10	
0.23.0	2011/11/17	
0.20.205.0	2011/10/17	
0.20.204.0	2011/9/5	
0.20.203.0	2011/5/11	
0.21.0	2010/8/23	
0.20.2	2010/2/26	
0.20.1	2009/9/14	
0.19.2	2009/7/23	
0.20.0	2009/4/22	
0.19.1	2009/2/24	
0.18.3	2009/1/29	
0.19.0	**2008/11/21**	
0.18.2	2008/11/3	
0.18.1	2008/9/17	
0.18.0	2008/8/22	
0.17.2	2008/8/19	
0.17.1	2008/6/23	
0.17.0	2008/5/20	
0.16.4	2008/5/5	
0.16.3	2008/4/16	

0.16.2	2008/4/2	
0.16.1	2008/3/13	
0.16.0	2008/2/7	
0.15.3	2008/1/18	
0.15.2	2008/1/2	
0.15.1	2007/11/27	
0.14.4	2007/11/26	
0.15.0	2007/10/29	
0.14.3	2007/10/19	
0.14.1	2007/9/4	

完整请浏览：http://hadoop.apache.org/releases.html。

3. 部署

3.1. 机器列表

共 5 台机器（zookeeper 部署在这 5 台机器上），部署如下表所示：

NameNode	JournalNode	DataNode	ZooKeeper
10.148.137.143 10.148.137.204	10.148.137.143 10.148.137.204 10.148.138.11	10.148.138.11 10.148.140.14 10.148.140.15	10.148.137.143 10.148.137.204 10.148.138.11 10.148.140.14 10.148.140.15

3.2. 主机名

机器 IP	对应的主机名
10.148.137.143	hadoop-137-143
10.148.137.204	hadoop-137-204
10.148.138.11	hadoop-138-11
10.148.140.14	hadoop-140-14
10.148.140.15	hadoop-140-15

注意主机名不能有下划线，否则启动时，SecondaryNameNode 节点会报如下所示的错误（取自 hadoop-hadoop-secondarynamenode-**VM_39_166_sles10_64**.out 文件）：

```
Java HotSpot(TM) 64-Bit Server VM warning: You have loaded library /data/hadoop/hadoop-2.7.2/lib/native/libhadoop.so.1.0.0 which might have disabled stack guard. The VM will try to fix the stack guard now.
It's highly recommended that you fix the library with 'execstack -c <libfile>', or link it with '-z noexecstack'.
Exception in thread "main" java.lang.IllegalArgumentException: The value of property bind.address must not be
```

```
null
        at com.google.common.base.Preconditions.checkArgument(Preconditions.java:88)
        at org.apache.hadoop.conf.Configuration.set(Configuration.java:971)
        at org.apache.hadoop.conf.Configuration.set(Configuration.java:953)
        at org.apache.hadoop.http.HttpServer2.initializeWebServer(HttpServer2.java:391)
        at org.apache.hadoop.http.HttpServer2.<init>(HttpServer2.java:344)
        at org.apache.hadoop.http.HttpServer2.<init>(HttpServer2.java:104)
        at org.apache.hadoop.http.HttpServer2$Builder.build(HttpServer2.java:292)
        at
org.apache.hadoop.hdfs.server.namenode.SecondaryNameNode.initialize(SecondaryNameNode.java:264)
        at
org.apache.hadoop.hdfs.server.namenode.SecondaryNameNode.<init>(SecondaryNameNode.java:192)
        at org.apache.hadoop.hdfs.server.namenode.SecondaryNameNode.main(SecondaryNameNode.java:651)
```

3.2.1.临时修改主机名

命令 hostname 不但可以查看主机名，还可以用它来修改主机名，格式为：hostname 新主机名。

在修改之前 172.25.40.171 对应的主机名为 VM-40-171-sles10-64，而 172.25.39.166 对应的主机名为 VM_39_166_sles10_64。两者的主机名均带有**下划线**，因此需要修改。为求简单，仅将原**下划线**改成**横线**：

hostname VM-40-171-sles10-64

hostname VM-39-166-sles10-64

经过上述修改后，还不够，类似于修改环境变量，还需要通过修改系统配置文件做永久修改。

3.2.2. 永久修改主机名

不同的 Linux 发行版本，对应的系统配置文件可能不同，SuSE 10.1 是/etc/**HOSTNAME**：

```
# cat /etc/HOSTNAME
VM_39_166_sles10_64
```

将文件中的“VM_39_166_sles10_64”，改成“VM-39-166-sles10-64”。有些 Linux 发行版本对应的可能是/etc/hostname 文件，有些可能是/etc/sysconfig/network 文件。

不但所在文件不同，修改的方法可能也不一样，比如有些是名字对形式，如：HOSTNAME=主机名。

修改之后，需要重启网卡，以使修改生效，执行命令：/etc/**rc.d/boot.localnet start**（不同系统命令会有差异，这是 SuSE 上的方法），再次使用 hostname 查看，会发现主机名变了。

直接重启系统，也可以使修改生效。

注意修改主机名后，需要重新验证 ssh 免密码登录，方法为：ssh 用户名@新的主机名。

3.3. 免密码登录范围

要求能通过免登录包括使用 IP 和主机名都能免密码登录：

1) NameNode 能免密码登录所有的 DataNode
2) 各 NameNode 能免密码登录自己
3) 各 NameNode 间能免密码互登录
4) DataNode 能免密码登录自己
5) DataNode 不需要配置免密码登录 NameNode 和其它 DataNode。

注：**免密码登录不是必须的**，如果不使用 hadoop-daemons.sh 等需要 ssh、scp 的脚本。

4. 约定

4.1. 安装目录约定

为便于讲解，本文约定 Hadoop、JDK 安装目录如下：

	安装目录	版本	说明
JDK	/data/**jdk**	1.7.0	ln -s /data/jdk1.7.0_55 /data/jdk
Hadoop	/data/**hadoop/hadoop**	2.7.2	ln -s /data/hadoop/hadoop-2.7.2 /data/hadoop/hadoop

在实际安装部署时，可以根据实际进行修改。

4.2. 服务端口约定

端口	作用
9000	fs.**defaultFS**，如：hdfs://172.25.40.171:9000
9001	dfs.namenode.**rpc**-address，DataNode 会连接这个端口
50070	dfs.**namenode**.**http**-address
50470	dfs.**namenode**.https-address
50100	dfs.namenode.**backup**.address
50105	dfs.namenode.**backup**.**http**-address
50090	dfs.namenode.**secondary**.**http**-address，如：172.25.39.166:50090
50091	dfs.namenode.**secondary**.https-address，如：172.25.39.166:50091
50020	dfs.**datanode**.**ipc**.address
50075	dfs.**datanode**.**http**.address
50475	dfs.**datanode**.https.address
50010	dfs.**datanode**.address，DataNode 的数据传输端口
8480	**dfs**.journalnode.rpc-address
8481	**dfs**.journalnode.https-address
8032	**yarn**.resourcemanager.address
8088	**yarn**.resourcemanager.**webapp**.address，YARN 的 http 端口
8090	**yarn**.resourcemanager.webapp.https.address
8030	**yarn**.resourcemanager.scheduler.address
8031	**yarn**.resourcemanager.resource-tracker.address
8033	**yarn**.resourcemanager.admin.address
8042	**yarn**.nodemanager.**webapp**.address
8040	**yarn**.nodemanager.localizer.address
8188	**yarn**.timeline-service.**webapp**.address
10020	**mapreduce**.jobhistory.address
19888	**mapreduce**.jobhistory.**webapp**.address
2888	**ZooKeeper**，如果是 Leader，用来监听 Follower 的连接
3888	**ZooKeeper**，用于 Leader 选举
2181	**ZooKeeper**，用来监听客户端的连接
16010	hbase.master.info.port，**HMaster** 的 **http** 端口
16000	hbase.master.port，**HMaster** 的 RPC 端口
60030	hbase.regionserver.info.port，**HRegionServer** 的 **http** 端口
60020	hbase.regionserver.port，**HRegionServer** 的 RPC 端口
8080	hbase.rest.port，**HBase** REST server 的端口
10000	**hive**.server2.thrift.port
9083	**hive**.metastore.uris

4.3. 各模块 RPC 和 HTTP 端口

模块	RPC 端口	HTTP 端口	HTTPS 端口
HDFS JournalNode	8485	8480	8481
HDFS NameNode	8020	50070	
HDFS DataNode	50020	50075	
HDFS SecondaryNameNode		50090	50091
Yarn Resource Manager	8032	8088	8090
Yarn Node Manager	8040	8042	
Yarn SharedCache		8788	

注：DataNode 通过端口 **50010** 传输数据。

5. 工作详单

为运行 Hadoop（HDFS、YARN 和 MapReduce）需要完成的工作详单：

JDK 安装	Hadoop 是 Java 语言开发的，所以需要。
免密码登录	NameNode 控制 SecondaryNameNode 和 DataNode 使用了 ssh 和 scp 命令，需要无密码执行。
Hadoop 安装和配置	这里指的是 HDFS、YARN 和 MapReduce，不包含 HBase、Hive 等的安装。

6. JDK 安装

本文安装的 JDK 1.7.0 版本。

6.1. 下载安装包

JDK 最新二进制安装包下载网址：
http://www.**oracle**.com/technetwork/java/**javase**/downloads
JDK1.7 二进制安装包下载网址：
http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html

本文下载的是 64 位 Linux 版本的 JDK1.7：jdk-7u55-linux-x64.gz。请不要安装 JDK1.8 版本，JDK1.8 和 Hadoop 2.7.2 不匹配，编译 Hadoop 2.7.2 源码时会报很多错误。

6.2. 安装步骤

JDK 的安装非常简单，将 jdk-7u55-linux-x64.gz 上传到 Linux，然后解压，接着配置好

环境变量即可（本文 jdk-7u55-linux-x64.gz 被上传在/data 目录下）：

1) 进入/data 目录
2) 解压安装包：tar xzf jdk-7u55-linux-x64.gz，解压后会在生成目录/data/jdk1.7.0_55
3) 建立软件链接：ln -s **/data/jdk1.7.0_55** /data/jdk
4) 修改/etc/profile 或用户目录下的 profile，或同等文件，配置如下所示环境变量：

```
export JAVA_HOME=/data/jdk
export CLASSPATH=$JAVA_HOME/lib/tools.jar
export PATH=$JAVA_HOME/bin:$PATH
```

完成这项操作之后，需要重新登录，或 source 一下 profile 文件，以便环境变量生效，当然也可以手工运行一下，以即时生效。如果还不放心，可以运行下 java 或 javac，看看命令是否可执行。如果在安装 JDK 之前，已经可执行了，则表示不用安装 JDK。

7. 免密码 ssh2 登录

以下针对的是 **ssh2，而不是 ssh，也不包括 OpenSSH**。配置分两部分：一是对**登录**机的配置，二是对**被登录**机的配置，其中**登录**机为客户端，**被登录**机为服务端，也就是解决客户端到服务端的无密码登录问题。下述涉及到的命令，可以直接拷贝到 Linux 终端上执行，已全部验证通过，操作环境为 SuSE 10.1。

第一步，修改所有**被登录**机上的 sshd 配置文件/etc/ssh2/**sshd2_config**：

1) （如果不以 root 用户运行 hadoop，则跳过这一步）将 **PermitRootLogin** 值设置为 yes，也就是取掉前面的注释号#
2) 将 **AllowedAuthentications** 值设置为 publickey,password，也就是取掉前面的注释号#
3) 重启 sshd 服务：**service** ssh2 restart

第二步，在所有**登录**机上，执行以下步骤：

1) 进入到.ssh2 目录：cd ~/.ssh2
2) **ssh-keygen2** -t dsa -P"

-P 表示密码，-P"就表示空密码，也可以不用-P 参数，但这样就要敲三次回车键，用-P"就一次回车。

成功之后，会在用户的主目录下生成私钥文件 **id_dsa_2048_a**，和公钥文件 **id_dsa_2048_a.pub**。

3) 生成 **identification** 文件：echo "IdKey **id_dsa_2048_a**" >> identification，请注意 IdKey 后面有一个空格，确保 identification 文件内容如下：

```
# cat identification
IdKey id_dsa_2048_a
```

4) 将文件 id_dsa_2048_a.pub，上传到所有**被登录**机的~/.ssh2 目录：scp **id_dsa_2048_a.pub** root@192.168.0.1:/root/.ssh2，这里假设 192.168.0.1 为其中一个**被登录**机的 IP。在执行 scp 之前，请确保 192.168.0.1 上有/root/.ssh2 这个目录，而/root/需要修改为 root 用户的实际 HOME 目录，通常环境变量$HOME 为用户主目录，~也表示用户主目录，不带任何参数的 cd 命令也会直接切换到用户主目录。

第三步，在所有**被登录**机上，执行以下步骤：

1) 进入到.ssh2 目录：cd ~/.ssh2
2) 生成 **authorization** 文件：echo "Key **id_dsa_2048_a.pub**" >> authorization，请注意 Key 后面有一个空格，确保 authorization 文件内容如下：

```
# cat authorization
Key id_dsa_2048_a.pub
```

完成上述工作之后，从**登录**机到**被登录**机的 ssh 登录就不需要密码了。如果没有配置好免密码登录，在启动时会遇到如下错误：

```
Starting namenodes on [172.25.40.171]
172.25.40.171: Host key not found from database.
172.25.40.171: Key fingerprint:
172.25.40.171: xofiz-zilip-tokar-rupyb-tufer-tahyc-sibah-kyvuf-palik-hazyt-duxux
172.25.40.171: You can get a public key's fingerprint by running
172.25.40.171: % ssh-keygen -F publickey.pub
172.25.40.171: on the keyfile.
172.25.40.171: warning: tcgetattr failed in ssh_rl_set_tty_modes_for_fd: fd 1: Invalid argument
```

或下列这样的错误：

```
Starting namenodes on [172.25.40.171]
172.25.40.171: hadoop's password:
```

建议生成的私钥和公钥文件名都带上自己的 IP，否则会有些混乱。

按照中**免密码登录范围**的说明，配置好所有的免密码登录。更多关于免密码登录说明，请浏览技术博客：

1) http://blog.chinaunix.net/uid-20682147-id-4212099.html（两个 SSH2 间免密码登录）
2) http://blog.chinaunix.net/uid-20682147-id-4212097.html（SSH2 免密码登录 OpenSSH）
3) http://blog.chinaunix.net/uid-20682147-id-4212094.html（OpenSSH 免密码登录 SSH2）
4) http://blog.chinaunix.net/uid-20682147-id-5520240.html（两个 openssh 间免密码登录）

8. Hadoop 安装和配置

本部分仅包括 HDFS、MapReduce 和 Yarn 的安装，不包括 HBase、Hive 等的安装。

8.1. 下载安装包

Hadoop 二进制安装包下载网址：http://hadoop.apache.org/releases.html#Download（或直接进入 http://mirror.bit.edu.cn/apache/hadoop/common/进行下载），本文下载的是 hadoop-2.7.2 版本（安装包：

http://www.apache.org/dyn/closer.cgi/hadoop/common/hadoop-2.7.2/hadoop-2.7.2.tar.gz，源码包：

http://www.apache.org/dyn/closer.cgi/hadoop/common/hadoop-2.7.2/hadoop-2.7.2-src.tar.gz），并不是稳定版本，最新的稳定版本是 hadoop-2.2.0。

官方的安装说明请浏览 Cluster Setup：
http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-common/ClusterSetup.html。

8.2. 安装和环境变量配置

1) 将 Hadoop 安装包 hadoop-2.7.2.tar.gz 上传到/data/hadoop 目录下
2) 进入/data/hadoop 目录
3) 在/data/hadoop 目录下，解压安装包 hadoop-2.7.2.tar.gz：tar xzf hadoop-2.7.2.tar.gz
4) 建立软件链接：ln -s /data/hadoop/**hadoop-2.7.2** /data/hadoop/**hadoop**
5) 修改用户主目录下的文件.profile（当然也可以是/etc/profile 或其它同等效果的文件），设置 Hadoop 环境变量：

```
export JAVA_HOME=/data/jdk
export HADOOP_HOME=/data/hadoop/hadoop
export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop
export PATH=$HADOOP_HOME/bin:$PATH
```

需要重新登录以生效，或者在终端上执行：export HADOOP_HOME=/data/hadoop/hadoop 也可以即时生效。

8.3. 修改 hadoop-env.sh

修改**所有节点**上的$HADOOP_HOME/etc/hadoop/hadoop-env.sh 文件，在靠近文件头部分加入：**export JAVA_HOME=/data/jdk**

特别说明一下：虽然在**/etc/profile** 已经添加了 JAVA_HOME，但仍然得修改所有节点上的 **hadoop-env.sh**，否则启动时，报如下所示的错误：

```
10.12.154.79: Error: JAVA_HOME is not set and could not be found.
10.12.154.77: Error: JAVA_HOME is not set and could not be found.
10.12.154.78: Error: JAVA_HOME is not set and could not be found.
10.12.154.78: Error: JAVA_HOME is not set and could not be found.
10.12.154.77: Error: JAVA_HOME is not set and could not be found.
10.12.154.79: Error: JAVA_HOME is not set and could not be found.
```

除 JAVA_HOME 之外，再添加：
export HADOOP_HOME=/data/hadoop/hadoop
export HADOOP_CONF_DIR=${HADOOP_HOME}/etc/hadoop

同时，建议将下列添加到/etc/**profile** 或~/**.profile** 中：
export JAVA_HOME=/data/jdk
export HADOOP_HOME=/data/hadoop/hadoop
export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop

8.4. 修改/etc/hosts

为省去不必要的麻烦，建议在所有节点的/etc/hosts 文件，都做如下所配置：

```
10.148.137.143 hadoop-137-143 # NameNode
10.148.137.204 hadoop-137-204 # NameNode
10.148.138.11  hadoop-138-11  # DataNode
10.148.140.14  hadoop-140-14  # DataNode
10.148.140.15  hadoop-140-15  # DataNode
```

注意不要为一个 IP 配置多个不同主机名，否则 HTTP 页面可能无法正常运作。

主机名，如 VM-39-166-sles10-64，可通过 hostname 命令取得。由于都配置了主机名，在启动 HDFS 或其它之前，需要确保**针对主机名进行过 ssh**，否则启动时，会遇到如下所示的错误：

```
VM-39-166-sles10-64: Host key not found from database.
VM-39-166-sles10-64: Key fingerprint:
VM-39-166-sles10-64: xofiz-zilip-tokar-rupyb-tufer-tahyc-sibah-kyvuf-palik-hazyt-duxux
VM-39-166-sles10-64: You can get a public key's fingerprint by running
VM-39-166-sles10-64: % ssh-keygen -F publickey.pub
VM-39-166-sles10-64: on the keyfile.
VM-39-166-sles10-64: warning: tcgetattr failed in ssh_rl_set_tty_modes_for_fd: fd 1: Invalid argument
```

上述错误表示没有以主机名 ssh 过一次 VM-39-166-sles10-64。按下列方法修复错误：

```
ssh hadoop@VM-39-166-sles10-64
Host key not found from database.
Key fingerprint:
xofiz-zilip-tokar-rupyb-tufer-tahyc-sibah-kyvuf-palik-hazyt-duxux
You can get a public key's fingerprint by running
% ssh-keygen -F publickey.pub
on the keyfile.
Are you sure you want to continue connecting (yes/no)? yes
Host key saved to /data/hadoop/.ssh2/hostkeys/key_36000_137vm_13739_137166_137sles10_13764.pub
host key for VM-39-166-sles10-64, accepted by hadoop Thu Apr 17 2014 12:44:32 +0800
Authentication successful.
Last login: Thu Apr 17 2014 09:24:54 +0800 from 10.32.73.69
Welcome to SuSE Linux 10 SP2 64Bit Nov 10,2010 by DIS
Version v2.6.20101110
No mail.
```

8.5. 修改 slaves

slaves 即为 HDFS 的 DataNode 节点。当使用脚本 start-dfs.sh 来启动 hdfs 时，会使用到这个文件，以无密码登录方式到各 slaves 上启动 DataNode。

修改主 NameNode 和备 NameNode 上的$HADOOP_HOME/etc/hadoop/slaves 文件，将

slaves 的节点 IP（也可以是相应的主机名）一个个加进去，一行一个 IP，如下所示：

```
> cat slaves
10.148.138.11
10.148.140.14
10.148.140.15
```

8.6. 准备好各配置文件

配置文件放在$HADOOP_HOME/etc/hadoop 目录下，对于 Hadoop 2.3.0、Hadoop 2.7.2 和 Hadoop 2.7.2 版本，该目录下的 core-site.xml、yarn-site.xml、hdfs-site.xml 和 mapred-site.xml 都是空的。如果不配置好就启动，如执行 start-dfs.sh，则会遇到各种错误。

可从$HADOOP_HOME/share/hadoop 目录下拷贝一份到/etc/hadoop 目录，然后在此基础上进行修改（以下内容可以直接拷贝执行，2.3.0 版本中各 default.xml 文件路径不同于 2.7.2 版本）：

```
# 进入$HADOOP_HOME 目录
cd $HADOOP_HOME
cp ./share/doc/hadoop/hadoop-project-dist/hadoop-common/core-default.xml ./etc/hadoop/core-site.xml
cp ./share/doc/hadoop/hadoop-project-dist/hadoop-hdfs/hdfs-default.xml ./etc/hadoop/hdfs-site.xml
cp ./share/doc/hadoop/hadoop-yarn/hadoop-yarn-common/yarn-default.xml ./etc/hadoop/yarn-site.xml
cp ./share/doc/hadoop/hadoop-mapreduce-client/hadoop-mapreduce-client-core/mapred-default.xml ./etc/hadoop/
mapred-site.xml
```

接下来，需要对默认的 core-site.xml、yarn-site.xml、hdfs-site.xml 和 mapred-site.xml 进行适当的修改，否则仍然无法启动成功。

QJM 的配置参照的官方文档：

http://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/HDFSHighAvailabilityWithQJM.html。

8.7. 修改 hdfs-site.xml

对 hdfs-site.xml 文件的修改，涉及下表中的属性：

属性名	属性值	说明
dfs.**nameservices**	**mycluster**	
dfs.ha.namenodes.**mycluster**	nn1,nn2	同一 nameservice 下，只能配置一或两个，也就是不能有 nn3 了
dfs.namenode.**rpc-address**.mycluster.nn1	hadoop-137-143:8020	
dfs.namenode.rpc-address.mycluster.nn2	hadoop-137-204:8020	
dfs.namenode.**http-address**.mycluster.nn1	hadoop-137-143:50070	
dfs.namenode.http-address.mycluster.nn2	hadoop-137-204:50070	
dfs.namenode.**shared.edits.dir**	qjournal://hadoop-137-14	至少三台 Quorum

	3:8485;hadoop-137-204:8485;hadoop-138-11:8485/mycluster	Journal 节点配置
dfs.client.**failover**.proxy.provider.mycluster	org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider	客户端通过它来找主 NameNode
dfs.ha.**fencing**.methods	sshfence 如果配置为 sshfence，当主 NameNode 异常时，使用 ssh 登录到主 NameNode，然后使用 fuser 将主 NameNode 杀死，因此需要确保所有 NameNode 上可以使用 fuser。	用来保证同一时刻只有一个主 NameNode，以防止脑裂。可带用户名和端口参数，格式示例：sshfence([[username][:port]])；值还可以为 shell 脚本，格式示例：shell(/path/to/my/script.sh arg1 arg2 ...)，如：shell(**/bin/true**) 如果 sshd 不是默认的 22 端口时，就需要指定。
dfs.ha.fencing.ssh.**private-key-files**	/data/hadoop/.ssh2/id_dsa_2048_a	指定私钥，如果是 OpenSSL，则值为 /data/hadoop/.ssh/id_rsa
dfs.ha.fencing.ssh.connect-**timeout**	30000	可选的配置
dfs.**journalnode**.edits.dir	/data/hadoop/hadoop/journal	JournalNode 存储其本地状态的位置，在 JouralNode 机器上的绝对路径，JNs 的 edits 和其他本地状态将被存储在此处
dfs.datanode.**data**.dir	/data/hadoop/hadoop/data	
dfs.namenode.**name**.dir		NameNode 元数据存放目录，默认值为 file://${hadoop.tmp.dir}/dfs/name，也就是在临时目录下，可以考虑放到数据目录下

dfs.namenode.**checkpoint**.dir		默认值为file://${hadoop.tmp.dir}/dfs/namesecondary，但如果没有启用SecondaryNameNode，则不需要
dfs.ha.**automatic-failover**.enabled	true	自动主备切换
dfs.datanode.max.xcievers	4096	可选修改，类似于linux的最大可打开的文件个数，默认为256，建议设置成大一点。同时，需要保证**系统可打开的文件个数**足够（可通过ulimit命令查看）。该错误会导致hbase报“notservingregionexception”。
dfs.journalnode.rpc-address	0.0.0.0:8485	配置JournalNode的RPC端口号，默认为0.0.0.0:8485，可以不用修改

详细配置可参考：
http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/hdfs-default.xml。

8.8. 修改 core-site.xml

对 core-site.xml 文件的修改，涉及下表中的属性：

属性名	**属性值**	**说明**
fs.defaultFS	**hdfs://mycluster**	
fs.default.name	hdfs://mycluster	按理应当不用填写这个参数，因为fs.defaultFS已取代它，但启动时报错：fs.defaultFS is file:///
hadoop.tmp.dir	/data/hadoop/hadoop/tmp	
ha.**zookeeper**.quorum	hadoop-137-143:2181,hadoop-138-11:2181,hadoop-140-14:2181	
ha.zookeeper.parent-**znode**	/mycluster/hadoop-ha	
io.seqfile.local.dir		默认值为${hadoop.tmp.dir}/io/local

fs.s3.buffer.dir		默认值为${hadoop.tmp.dir}/s3
fs.s3a.buffer.dir		默认值为${hadoop.tmp.dir}/s3a

注意启动之前，需要将配置的目录创建好，如创建好/data/hadoop/current/tmp 目录。详细可参考：
http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-common/core-default.xm。

8.8.1. dfs.namenode.rpc-address

如果没有配置，则启动时报如下错误：

Incorrect configuration: namenode address dfs.namenode.servicerpc-address or dfs.namenode.rpc-address is not configured.

这里需要指定 IP 和端口，如果只指定了 IP，如<value>10.148.137.143</value>，则启动时输出如下：

Starting namenodes on []

改成“<value>**hadoop-137-143:8020**</value>”后，则启动时输出为：

Starting namenodes on [**10.148.137.143**]

8.9. 修改 mapred-site.xml

对 hdfs-site.xml 文件的修改，涉及下表中的属性：

属性名	**属性值**	**涉及范围**
mapreduce.framework.name	**yarn**	所有 mapreduce 节点

详细配置可参考：
http://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/mapred-default.xml。

8.10. 修改 yarn-site.xml

对 yarn-site.xml 文件的修改，涉及下表中的属性：

属性名	**属性值**	**涉及范围**
~~yarn.resourcemanager.hostname~~	~~0.0.0.0~~	~~ResourceManager~~ ~~NodeManager~~ **HA 模式可不配置，但由于其它配置项可能有引用它，建议保持值为 0.0.0.0，如果没有被引用到，则可不配置。**

yarn.nodemanager.hostname	0.0.0.0	
yarn.nodemanager.aux-services	mapreduce_shuffle	
以下为 HA 相关的配置，包括自动切换（可仅可在 ResourceManager 节点上配置）		
yarn.resourcemanager.ha.enabled	true	启用 HA
yarn.resourcemanager.cluster-id	yarn-cluster	可不同于 HDFS 的
yarn.resourcemanager.ha.rm-ids	rm1,rm2	注意 NodeManager 要和 ResourceManager 一样配置
yarn.resourcemanager.hostname.rm1	hadoop-137-143	
yarn.resourcemanager.hostname.rm2	hadoop-137-204	
yarn.resourcemanager.webapp.address.rm1	hadoop-137-143:8088	
yarn.resourcemanager.webapp.address.rm2	hadoop-137-204:8088	
yarn.resourcemanager.zk-address	hadoop-137-143:2181,hadoop-137-204:2181,hadoop-138-11:2181	
yarn.resourcemanager.ha.automatic-failover.enable	true	可不配置，因为当 yarn.resourcemanager.ha.enabled 为 true 时，它的默认值即为 true

yarn.nodemanager.hostname 如果配置成具体的 IP，如 10.12.154.79，则会导致每个 NodeManager 的配置不同。详细配置可参考：

http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-common/yarn-default.xml。Yarn HA 的配置可以参考：

https://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/ResourceManagerHA.html。

9. 启动顺序

Zookeeper -> JournalNode -> **格式化 NameNode** -> **初始化 JournalNode** -> **创建命名空间（zkfc）** -> NameNode -> DataNode -> ResourceManager -> NodeManager。但请注意首次启动 NameNode 之前，得先做 format，也请注意备 NameNode 的启动方法。

10. 启动 HDFS

在启动 HDFS 之前，需要先完成对 NameNode 的格式化。

10.1. 创建好目录

mkdir -p /data/hadoop/hadoop-2.7.2/tmp/dfs/name

10.2. 启动好 zookeeper

./zkServer.sh start
注意在启动其它之前先启动 zookeeper。

10.3. 创建命名空间

在其中一个 namenode 上执行：
./hdfs zkfc -formatZK

注意，这一步要在初始化命名空间之后执行。

10.4. 启动所有 JournalNode

NameNode 将元数据操作日志记录在 JournalNode 上，主备 NameNode 通过记录在 JouralNode 上的日志完成元数据同步。

在所有 JournalNode 上执行（注意是两个参数，在“**hdfs** namenode -format”之后做这一步）：
./**hadoop-daemon**.sh **start journalnode**

注意，在执行“**hdfs** namenode -format”之前，必须先启动好 JournalNode，而 format 又必须在启动 namenode 之前。

10.5. 初始化 JournalNode

如果是非 HA 转 HA 才需要这一步，在其中一个 JournalNode 上执行：
./hdfs namenode **-initializeSharedEdits**

此命令默认是交互式的，加上参数-force 转成非交互式。

在所有 JournalNode 创建如下目录：
mkdir -p /data/hadoop/hadoop/journal/mycluster/current

10.6. 格式化 NameNode

注意只有新的，才需要做这一步，而且只需要在主 NameNode 上执行。

1) 进入$HADOOP_HOME/**bin** 目录
2) 进行格式化：./**hdfs** namenode -format
   如果完成有，输出包含“INFO util.ExitUtil: Exiting with status 0”，则表示格式化成功。
   在进行格式化时，如果没有在/etc/hosts 文件中添加主机名和 IP 的映射：“172.25.40.171

VM-40-171-sles10-64”，则会报如下所示错误：

```
14/04/17 03:44:09 WARN net.DNS: Unable to determine local hostname -falling back to "localhost"
java.net.UnknownHostException: VM-40-171-sles10-64: VM-40-171-sles10-64: unknown error
        at java.net.InetAddress.getLocalHost(InetAddress.java:1484)
        at org.apache.hadoop.net.DNS.resolveLocalHostname(DNS.java:264)
        at org.apache.hadoop.net.DNS.<clinit>(DNS.java:57)
        at org.apache.hadoop.hdfs.server.namenode.NNStorage.newBlockPoolID(NNStorage.java:945)
        at org.apache.hadoop.hdfs.server.namenode.NNStorage.newNamespaceInfo(NNStorage.java:573)
        at org.apache.hadoop.hdfs.server.namenode.FSImage.format(FSImage.java:144)
        at org.apache.hadoop.hdfs.server.namenode.NameNode.format(NameNode.java:845)
        at org.apache.hadoop.hdfs.server.namenode.NameNode.createNameNode(NameNode.java:1256)
        at org.apache.hadoop.hdfs.server.namenode.NameNode.main(NameNode.java:1370)
Caused by: java.net.UnknownHostException: VM-40-171-sles10-64: unknown error
        at java.net.Inet4AddressImpl.lookupAllHostAddr(Native Method)
        at java.net.InetAddress$2.lookupAllHostAddr(InetAddress.java:907)
        at java.net.InetAddress.getAddressesFromNameService(InetAddress.java:1302)
        at java.net.InetAddress.getLocalHost(InetAddress.java:1479)
        ... 8 more
```

10.7. 启动主 NameNode

1) 进入$HADOOP_HOME/**sbin** 目录
2) 启动主 NameNode：

./hadoop-daemon.sh start **namenode**

启动时，遇到如下所示的错误，则表示 NameNode 不能免密码登录自己。如果之前使用 IP 可以免密码登录自己，则原因一般是因为没有使用主机名登录过自己，因此解决办法是使用主机名 SSH 一下，比如：ssh hadoop@**VM_40_171_sles10_64**，然后再启动。

```
Starting namenodes on [VM_40_171_sles10_64]
VM_40_171_sles10_64: Host key not found from database.
VM_40_171_sles10_64: Key fingerprint:
VM_40_171_sles10_64: xofiz-zilip-tokar-rupyb-tufer-tahyc-sibah-kyvuf-palik-hazyt-duxux
VM_40_171_sles10_64: You can get a public key's fingerprint by running
VM_40_171_sles10_64: % ssh-keygen -F publickey.pub
VM_40_171_sles10_64: on the keyfile.
VM_40_171_sles10_64: warning: tcgetattr failed in ssh_rl_set_tty_modes_for_fd: fd 1: Invalid argument
```

10.8. 启动备 NameNode

1) ./hdfs namenode **-bootstrapStandby**
2) ./hadoop-daemon.sh start **namenode**

如果没有执行第 1 步，直接启动会遇到如下错误：

No valid image files found
或者在该 NameNode 日志会发现如下错误：
2016-04-08 14:08:39,745 WARN org.apache.hadoop.hdfs.server.namenode.FSNamesystem: Encountered exception loading fsimage
java.io.IOException: **NameNode is not formatted**.

10.9. 启动主备切换进程

在所有 NameNode 上启动主备切换进程：
./hadoop-daemon.sh start **zkfc**
只有启动了 DFSZKFailoverController 进程，HDFS 才能自动切换主备。

注：**zkfc** 是 zookeeper failover controller 的缩写。

10.10. 启动所有 DataNode

在各个 DataNode 上分别执行：
./hadoop-daemon.sh start **datanode**

如果有发现 DataNode 进程并没有起来，可以试试删除 logs 目录下的 DataNode 日志，再得启看看。

10.11. 检查启动是否成功

1) 使用 JDK 提供的 jps 命令，查看相应的进程是否已启动
2) 检查$HADOOP_HOME/logs 目录下的 log 和 out 文件，看看是否有异常信息。

启动后 nn1 和 nn2 都处于备机状态，将 nn1 切换为主机：
./hdfs haadmin -transitionToActive nn1

10.11.1. DataNode

执行 jps 命令（注：jps 是 jdk 中的一个命令，不是 jre 中的命令），可看到 DataNode 进程：

```
$ jps
18669 DataNode
24542 Jps
```

10.11.2. NameNode

执行 jps 命令，可看到 NameNode 进程：

```
$ jps
18669 NameNode
24542 Jps
```

10.12. 执行 HDFS 命令

执行 HDFS 命令，以进一步检验是否已经安装成功和配置好。关于 HDFS 命令的用法，直接运行命令 hdfs 或 hdfs dfs，即可看到相关的用法说明。

10.12.1. 查看 DataNode 是否正常启动

hdfs dfsadmin -report

注意如果 core-site.xml 中的配置项 fs.default.name 的值为 file:///，则会报：
report: FileSystem file:/// is not an HDFS file system
Usage: hdfs dfsadmin [-report] [-live] [-dead] [-decommissioning]

解决这个问题，只需要将 fs.default.name 的值设置为和 fs.defaultFS 相同的值。

10.12.2. 查看 NameNode 的主备状态

如查看 NameNode1 和 NameNode2 分别是主还是备：

```
$ hdfs haadmin -getServiceState nm1
standby
$ hdfs haadmin -getServiceState nm2
active
```

10.12.3. hdfs dfs ls

“**hdfs dfs -ls**”带一个参数，如果参数以“hdfs://**URI**”打头表示访问 HDFS，否则相当于 ls。其中 URI 为 NameNode 的 IP 或主机名，可以包含端口号，即 hdfs-site.xml 中“dfs.namenode.rpc-address”指定的值。

“hdfs dfs -ls”要求默认端口为 8020，如果配置成 9000，则需要指定端口号，否则不用指定端口，这一点类似于浏览器访问一个 URL。示例：

```
> hdfs dfs -ls hdfs://172.25.40.171:9001/
```

9001 后面的斜杠/是和必须的，否则被当作文件。如果不指定端口号 9001，则使用默认的 8020，“172.25.40.171:9001”由 hdfs-site.xml 中“dfs.namenode.rpc-address”指定。

不难看出“**hdfs dfs -ls**”可以操作不同的 HDFS 集群，只需要指定不同的 **URI**。

文件上传后，被存储在 DataNode 的 data 目录下（由 DataNode 的 hdfs-site.xml 中的属性“**dfs.datanode.data.dir**”指定），如：

$HADOOP_HOME/data/current/BP-139798373-172.25.40.171-1397735615751/current/finalized/**blk_1073741825**

文件名中的“**blk**”是 block，即块的意思，默认情况下 **blk_1073741825** 即为文件的一个完整块，Hadoop 未对它进额外处理。

10.12.4. hdfs dfs -put

上传文件命令，示例：

```
> hdfs dfs -put /etc/SuSE-release hdfs://172.25.40.171:9001/
```

10.12.5. hdfs dfs -rm

删除文件命令，示例：

```
> hdfs dfs -rm hdfs://172.25.40.171:9001/SuSE-release
Deleted hdfs://172.25.40.171:9001/SuSE-release
```

10.12.6. 新 NameNode 如何加入？

当有 NameNode 机器损坏时，必然存在新 NameNode 来替代。把配置修改成指向新 NameNode，然后以备机形式启动新 NameNode，这样新的 NameNode 即加入到 Cluster 中：

```
1) ./hdfs namenode -bootstrapStandby
2) ./hadoop-daemon.sh start namenode
```

10.12.7. HDFS 只允许有一主一备两个 NameNode

如果试图配置三个 NameNode，如：

```
<property>
  <name>dfs.ha.namenodes.test</name>
  <value>nm1,nm2,nm3</value>
  <description>
    The prefix for a given nameservice, contains a comma-separated
    list of namenodes for a given nameservice (eg EXAMPLENAMESERVICE).
  </description>
</property>
```

则运行“hdfs namenode -bootstrapStandby”时会报如下错误，表示在同一 NameSpace 内不能超过 2 个 NameNode：

```
16/04/11 09:51:57 ERROR namenode.NameNode: Failed to start namenode.
java.io.IOException: java.lang.IllegalArgumentException: Expected exactly 2 NameNodes in namespace 'test'.
Instead, got only 3 (NN ids were 'nm1','nm2','nm3'
```

```
        at org.apache.hadoop.hdfs.server.namenode.ha.BootstrapStandby.run(BootstrapStandby.java:425)
        at org.apache.hadoop.hdfs.server.namenode.NameNode.createNameNode(NameNode.java:1454)
        at org.apache.hadoop.hdfs.server.namenode.NameNode.main(NameNode.java:1554)
Caused by: java.lang.IllegalArgumentException: Expected exactly 2 NameNodes in namespace 'test'. Instead, got only 3 (NN ids were 'nm1','nm2','nm3'
        at com.google.common.base.Preconditions.checkArgument(Preconditions.java:115)
```

10.12.8. 存储均衡 start-balancer.sh

示例：start-balancer.sh -t 10%
10%表示机器与机器之间磁盘使用率偏差小于 10%时认为均衡，否则做均衡搬动。
“start-balancer.sh”调用“hdfs start balancer”来做均衡，可以调用 stop-balancer.sh 停止均衡。

均衡过程非常慢，但是均衡过程中，仍能够正常访问 HDFS，包括往 HDFS 上传文件。

```
[VM2016@hadoop-030 /data4/hadoop/sbin]$ hdfs balancer # 可以改为调用 start-balancer.sh
16/04/08 14:26:55 INFO balancer.Balancer: namenodes  = [hdfs://test] // test 为 HDFS 的 cluster 名
16/04/08 14:26:55 INFO balancer.Balancer: parameters = Balancer.Parameters[BalancingPolicy.Node, threshold=10.0, max idle iteration = 5, number of nodes to be excluded = 0, number of nodes to be included = 0]
Time Stamp               Iteration#  Bytes Already Moved  Bytes Left To Move  Bytes Being Moved
16/04/08 14:26:56 INFO net.NetworkTopology: Adding a new node: /default-rack/192.168.1.231:50010
16/04/08 14:26:56 INFO net.NetworkTopology: Adding a new node: /default-rack/192.168.1.229:50010
16/04/08 14:26:56 INFO net.NetworkTopology: Adding a new node: /default-rack/192.168.1.213:50010
16/04/08 14:26:56 INFO net.NetworkTopology: Adding a new node: /default-rack/192.168.1.208:50010
16/04/08 14:26:56 INFO net.NetworkTopology: Adding a new node: /default-rack/192.168.1.232:50010
16/04/08 14:26:56 INFO net.NetworkTopology: Adding a new node: /default-rack/192.168.1.207:50010
16/04/08 14:26:56 INFO balancer.Balancer: 5 over-utilized: [192.168.1.231:50010:DISK, 192.168.1.229:50010:DISK, 192.168.1.213:50010:DISK, 192.168.1.208:50010:DISK, 192.168.1.232:50010:DISK]
16/04/08 14:26:56 INFO balancer.Balancer: 1 underutilized（未充分利用的）: [192.168.1.207:50010:DISK] # 数据将移向该节点
16/04/08 14:26:56 INFO balancer.Balancer: Need to move 816.01 GB to make the cluster balanced. # 需要移动 816.01G 数据达到平衡
16/04/08 14:26:56 INFO balancer.Balancer: Decided to move 10 GB bytes from 192.168.1.231:50010:DISK to 192.168.1.207:50010:DISK # 从 192.168.1.231 移动 10G 数据到 192.168.1.207
16/04/08 14:26:56 INFO balancer.Balancer: Will move 10 GB in this iteration

16/04/08 14:32:58 INFO balancer.Dispatcher: Successfully moved blk_1073749366_8542 with size=77829046 from 192.168.1.231:50010:DISK to 192.168.1.207:50010:DISK through 192.168.1.213:50010
16/04/08 14:32:59 INFO balancer.Dispatcher: Successfully moved blk_1073749386_8562 with size=77829046 from 192.168.1.231:50010:DISK to 192.168.1.207:50010:DISK through 192.168.1.231:50010
16/04/08 14:33:34 INFO balancer.Dispatcher: Successfully moved blk_1073749378_8554 with size=77829046 from 192.168.1.231:50010:DISK to 192.168.1.207:50010:DISK through 192.168.1.231:50010
16/04/08 14:34:38 INFO balancer.Dispatcher: Successfully moved blk_1073749371_8547 with size=134217728 from 192.168.1.231:50010:DISK to 192.168.1.207:50010:DISK through 192.168.1.213:50010
```

```
16/04/08 14:34:54 INFO balancer.Dispatcher: Successfully moved blk_1073749395_8571 with size=134217728 from
192.168.1.231:50010:DISK to 192.168.1.207:50010:DISK through 192.168.1.231:50010
Apr 8, 2016 2:35:01 PM            0            478.67 MB           816.01 GB              10 GB
16/04/08 14:35:10 INFO net.NetworkTopology: Adding a new node: /default-rack/192.168.1.213:50010
16/04/08 14:35:10 INFO net.NetworkTopology: Adding a new node: /default-rack/192.168.1.229:50010
16/04/08 14:35:10 INFO net.NetworkTopology: Adding a new node: /default-rack/192.168.1.232:50010
16/04/08 14:35:10 INFO net.NetworkTopology: Adding a new node: /default-rack/192.168.1.231:50010
16/04/08 14:35:10 INFO net.NetworkTopology: Adding a new node: /default-rack/192.168.1.208:50010
16/04/08 14:35:10 INFO net.NetworkTopology: Adding a new node: /default-rack/192.168.1.207:50010
16/04/08 14:35:10 INFO balancer.Balancer: 5 over-utilized: [192.168.1.213:50010:DISK, 192.168.1.229:50010:DISK,
192.168.1.232:50010:DISK, 192.168.1.231:50010:DISK, 192.168.1.208:50010:DISK]
16/04/08 14:35:10 INFO balancer.Balancer: 1 underutilized（未充分利用的）: [192.168.1.207:50010:DISK]
16/04/08 14:35:10 INFO balancer.Balancer: Need to move 815.45 GB to make the cluster balanced.
16/04/08 14:35:10 INFO balancer.Balancer: Decided to move 10 GB bytes from 192.168.1.213:50010:DISK to
192.168.1.207:50010:DISK
16/04/08 14:35:10 INFO balancer.Balancer: Will move 10 GB in this iteration

16/04/08 14:41:18 INFO balancer.Dispatcher: Successfully moved blk_1073760371_19547 with size=77829046 from
192.168.1.213:50010:DISK to 192.168.1.207:50010:DISK through 192.168.1.213:50010
16/04/08 14:41:19 INFO balancer.Dispatcher: Successfully moved blk_1073760385_19561 with size=77829046 from
192.168.1.213:50010:DISK to 192.168.1.207:50010:DISK through 192.168.1.213:50010
16/04/08 14:41:22 INFO balancer.Dispatcher: Successfully moved blk_1073760393_19569 with size=77829046 from
192.168.1.213:50010:DISK to 192.168.1.207:50010:DISK through 192.168.1.213:50010
16/04/08 14:41:23 INFO balancer.Dispatcher: Successfully moved blk_1073760363_19539 with size=77829046 from
192.168.1.213:50010:DISK to 192.168.1.207:50010:DISK through 192.168.1.213:50010
```

11. 启动 YARN

11.1. 启动 YARN

1) 进入$HADOOP_HOME/**sbin** 目录
2) 在主备两台都执行：start-yarn.sh，即开始启动 YARN

若启动成功，则在 Master 节点执行 jps，可以看到 **ResourceManager**：

```
> jps
24689 NameNode
30156 Jps
28861 ResourceManager
```

在 Slaves 节点执行 jps，可以看到 **NodeManager**：

```
$ jps
```

```
14019 NodeManager
23257 DataNode
15115 Jps
```

如果只需要单独启动指定节点上的 ResourceManager，这样：
./yarn-daemon.sh start resourcemanager

对于 NodeManager，则是这样：
./yarn-daemon.sh start nodemanager

11.2. 执行 YARN 命令

11.2.1. yarn node -list

列举 YARN 集群中的所有 NodeManager，如（**注意参数间的空格，直接执行 yarn 可以看到使用帮助**）：

```
> yarn node -list
Total Nodes:3
         Node-Id         Node-State Node-Http-Address       Number-of-Running-Containers
 localhost:45980           RUNNING       localhost:8042                                  0
 localhost:47551           RUNNING       localhost:8042                                  0
 localhost:58394           RUNNING       localhost:8042                                  0
```

11.2.2. yarn node -status

查看指定 NodeManager 的状态，如：

```
> yarn node -status localhost:47551
Node Report :
        Node-Id : localhost:47551
        Rack : /default-rack
        Node-State : RUNNING
        Node-Http-Address : localhost:8042
        Last-Health-Update : 星期五 18/四月/14 01:45:41:555GMT
        Health-Report :
        Containers : 0
        Memory-Used : 0MB
        Memory-Capacity : 8192MB
        CPU-Used : 0 vcores
        CPU-Capacity : 8 vcores
```

11.2.3. yarn rmadmin -getServiceState rm1

查看 rm1 的主备状态，即查看它是主（active）还是备（standby）。

11.2.4. yarn rmadmin -transitionToStandby rm1

将 rm1 从主切为备。
更多的 yarn 命令可以参考：
https://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YarnCommands.html。

12. 运行 MapReduce 程序

在安装目录的 share/hadoop/mapreduce 子目录下，有现存的示例程序：

```
hadoop@VM-40-171-sles10-64:~/hadoop> ls share/hadoop/mapreduce
hadoop-mapreduce-client-app-2.7.2.jar
hadoop-mapreduce-client-jobclient-2.7.2-tests.jar
hadoop-mapreduce-client-common-2.7.2.jar      hadoop-mapreduce-client-shuffle-2.7.2.jar
hadoop-mapreduce-client-core-2.7.2.jar        hadoop-mapreduce-examples-2.7.2.jar
hadoop-mapreduce-client-hs-2.7.2.jar          lib
hadoop-mapreduce-client-hs-plugins-2.7.2.jar  lib-examples
hadoop-mapreduce-client-jobclient-2.7.2.jar   sources
```

跑一个示例程序试试：

```
hdfs dfs -put /etc/hosts  hdfs://test/in/
hadoop jar ./share/hadoop/mapreduce/hadoop-mapreduce-examples-2.7.2.jar wordcount hdfs://test/in/ hdfs://test/out/
```

运行过程中，使用 java 的 jps 命令，可以看到 yarn 启动了名为 **YarnChild** 的进程。

wordcount 运行完成后，结果会保存在 out 目录下，保存结果的文件名类似于“part-r-00000”。另外，跑这个示例程序有两个需求注意的点：

1) in 目录下要有文本文件，或 in 即为被统计的文本文件，可以为 HDFS 上的文件或目录，也可以为本地文件或目录
2) out 目录不能存在，程序会自动去创建它，如果已经存在则会报错。

包 hadoop-mapreduce-**examples**-2.7.2.jar 中含有多个示例程序，不带参数运行，即可看到用法：

```
> hadoop jar ./share/hadoop/mapreduce/hadoop-mapreduce-examples-2.7.2.jar wordcount
Usage: wordcount <in> <out>

> hadoop jar ./share/hadoop/mapreduce/hadoop-mapreduce-examples-2.7.2.jar
An example program must be given as the first argument.
```

Valid program names are:

aggregatewordcount: An Aggregate based map/reduce program that counts the words in the input files.

aggregatewordhist: An Aggregate based map/reduce program that computes the histogram of the words in the input files.

bbp: A map/reduce program that uses Bailey-Borwein-Plouffe to compute exact digits of Pi.

dbcount: An example job that count the pageview counts from a database.

distbbp: A map/reduce program that uses a BBP-type formula to compute exact bits of Pi.

grep: A map/reduce program that counts the matches of a regex in the input.

join: A job that effects a join over sorted, equally partitioned datasets

multifilewc: A job that counts words from several files.

pentomino: A map/reduce tile laying program to find solutions to pentomino problems.

pi: A map/reduce program that estimates Pi using a quasi-Monte Carlo method.

randomtextwriter: A map/reduce program that writes 10GB of random textual data per node.

randomwriter: A map/reduce program that writes 10GB of random data per node.

secondarysort: An example defining a secondary sort to the reduce.

sort: A map/reduce program that sorts the data written by the random writer.

sudoku: A sudoku solver.

teragen: Generate data for the terasort

terasort: Run the terasort

teravalidate: Checking results of terasort

wordcount: A map/reduce program that counts the words in the input files.

wordmean: A map/reduce program that counts the average length of the words in the input files.

wordmedian: A map/reduce program that counts the median length of the words in the input files.

wordstandarddeviation: A map/reduce program that counts the standard deviation of the length of the words in the input files.

修改日志级别为 DEBBUG，并打屏：

export **HADOOP_ROOT_LOGGER**=DEBUG,console

13. 常见错误

13.1. 执行“hdfs dfs -ls”时报 ConnectException

原因可能是指定的端口号 **9000** 不对，该端口号由 hdfs-site.xml 中的属性“**dfs.namenode.rpc-address**”指定，即为 NameNode 的 RPC 服务端口号。

文件上传后，被存储在 DataNode 的 data（由 DataNode 的 hdfs-site.xml 中的属性“**dfs.datanode.data.dir**”指定）目录下，如：

$HADOOP_HOME/data/current/BP-139798373-172.25.40.171-1397735615751/current/finalized/**blk_1073741825**

文件名中的“**blk**”是 block，即块的意思，默认情况下 **blk_1073741825** 即为文件的一个完整块，Hadoop 未对它进额外处理。

```
hdfs dfs -ls hdfs://172.25.40.171:9000
14/04/17 12:04:02 WARN conf.Configuration: mapred-site.xml:an attempt to override final parameter: mapreduce.job.end-notification.max.attempts;  Ignoring.
14/04/17 12:04:02 WARN conf.Configuration: mapred-site.xml:an attempt to override final parameter: mapreduce.job.end-notification.max.retry.interval;  Ignoring.
14/04/17 12:04:02 WARN conf.Configuration: mapred-site.xml:an attempt to override final parameter: mapreduce.job.end-notification.max.attempts;  Ignoring.
14/04/17 12:04:02 WARN conf.Configuration: mapred-site.xml:an attempt to override final parameter: mapreduce.job.end-notification.max.retry.interval;  Ignoring.
14/04/17 12:04:02 WARN conf.Configuration: mapred-site.xml:an attempt to override final parameter: mapreduce.job.end-notification.max.attempts;  Ignoring.
14/04/17 12:04:02 WARN conf.Configuration: mapred-site.xml:an attempt to override final parameter: mapreduce.job.end-notification.max.retry.interval;  Ignoring.
Java HotSpot(TM) 64-Bit Server VM warning: You have loaded library /data/hadoop/hadoop-2.7.2/lib/native/libhadoop.so.1.0.0 which might have disabled stack guard. The VM will try to fix the stack guard now.
It's highly recommended that you fix the library with 'execstack -c <libfile>', or link it with '-z noexecstack'.
14/04/17 12:04:02 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
14/04/17 12:04:03 WARN conf.Configuration: mapred-site.xml:an attempt to override final parameter: mapreduce.job.end-notification.max.attempts;  Ignoring.
14/04/17 12:04:03 WARN conf.Configuration: mapred-site.xml:an attempt to override final parameter: mapreduce.job.end-notification.max.retry.interval;  Ignoring.
ls: Call From VM-40-171-sles10-64/172.25.40.171 to VM-40-171-sles10-64:9000 failed on connection exception: java.net.ConnectException: 拒绝连接; For more details see: http://wiki.apache.org/hadoop/ConnectionRefused
```

13.2. Initialization failed for Block pool

可能是因为对 NameNode 做 format 之前，没有清空 DataNode 的 data 目录。

13.3. Incompatible clusterIDs

“Incompatible clusterIDs”的错误原因是在执行“hdfs namenode -format”之前，没有清空 DataNode 节点的 data 目录。

网上一些文章和帖子说是 tmp 目录，它本身也是没问题的，但 Hadoop **2.7.2** 是 data 目录，实际上这个信息已经由日志的“**/data/hadoop/hadoop-2.7.2/data**”指出，所以不能死死的参照网上的解决办法，遇到问题时多仔细观察。

从上述描述不难看出，解决办法就是清空所有 DataNode 的 data 目录，但注意不要将 data

目录本身给删除了。

data 目录由 **core-site.xml** 文件中的属性“**dfs.datanode.data.dir**”指定。

2014-04-17 19:30:33,075 INFO org.apache.hadoop.hdfs.server.common.Storage: Lock on /data/hadoop/hadoop-2.7.2/data/in_use.lock acquired by nodename 28326@localhost
2014-04-17 19:30:33,078 FATAL org.apache.hadoop.hdfs.server.datanode.DataNode: Initialization failed for block pool Block pool <registering> (Datanode Uuid unassigned) service to /172.25.40.171:9001
java.io.IOException: **Incompatible clusterIDs** in **/data/hadoop/hadoop-2.7.2/data**: namenode clusterID = **CID-50401d89-a33e-47bf-9d14-914d8f1c4862**; datanode clusterID = **CID-153d6fcb-d037-4156-b63a-10d6be224091**
at org.apache.hadoop.hdfs.server.datanode.DataStorage.doTransition(DataStorage.java:472)
at org.apache.hadoop.hdfs.server.datanode.DataStorage.recoverTransitionRead(DataStorage.java:225)
at org.apache.hadoop.hdfs.server.datanode.DataStorage.recoverTransitionRead(DataStorage.java:249)
at org.apache.hadoop.hdfs.server.datanode.DataNode.initStorage(DataNode.java:929)
at org.apache.hadoop.hdfs.server.datanode.DataNode.initBlockPool(DataNode.java:900)
at org.apache.hadoop.hdfs.server.datanode.BPOfferService.verifyAndSetNamespaceInfo(BPOfferService.java:274)
at org.apache.hadoop.hdfs.server.datanode.BPServiceActor.connectToNNAndHandshake(BPServiceActor.java:220)
at org.apache.hadoop.hdfs.server.datanode.BPServiceActor.run(BPServiceActor.java:815)
at java.lang.Thread.run(Thread.java:744)
2014-04-17 19:30:33,081 WARN org.apache.hadoop.hdfs.server.datanode.DataNode: Ending block pool service for: Block pool <registering> (Datanode Uuid unassigned) service to /172.25.40.171:9001
2014-04-17 19:30:33,184 WARN org.apache.hadoop.hdfs.server.datanode.DataNode: Block pool ID needed, but service not yet registered with NN
java.lang.Exception: trace
at org.apache.hadoop.hdfs.server.datanode.BPOfferService.getBlockPoolId(BPOfferService.java:143)
at org.apache.hadoop.hdfs.server.datanode.BlockPoolManager.remove(BlockPoolManager.java:91)
at org.apache.hadoop.hdfs.server.datanode.DataNode.shutdownBlockPool(DataNode.java:859)
at

```
org.apache.hadoop.hdfs.server.datanode.BPOfferService.shutdownActor(BPOfferService.java:350
)
        at
org.apache.hadoop.hdfs.server.datanode.BPServiceActor.cleanUp(BPServiceActor.java:619)
        at
org.apache.hadoop.hdfs.server.datanode.BPServiceActor.run(BPServiceActor.java:837)
        at java.lang.Thread.run(Thread.java:744)
2014-04-17 19:30:33,184 INFO org.apache.hadoop.hdfs.server.datanode.DataNode: Removed
Block pool <registering> (Datanode Uuid unassigned)
2014-04-17 19:30:33,184 WARN org.apache.hadoop.hdfs.server.datanode.DataNode: Block pool
ID needed, but service not yet registered with NN
java.lang.Exception: trace
        at
org.apache.hadoop.hdfs.server.datanode.BPOfferService.getBlockPoolId(BPOfferService.java:143
)
        at
org.apache.hadoop.hdfs.server.datanode.DataNode.shutdownBlockPool(DataNode.java:861)
        at
org.apache.hadoop.hdfs.server.datanode.BPOfferService.shutdownActor(BPOfferService.java:350
)
        at
org.apache.hadoop.hdfs.server.datanode.BPServiceActor.cleanUp(BPServiceActor.java:619)
        at
org.apache.hadoop.hdfs.server.datanode.BPServiceActor.run(BPServiceActor.java:837)
        at java.lang.Thread.run(Thread.java:744)
2014-04-17 19:30:35,185 WARN org.apache.hadoop.hdfs.server.datanode.DataNode: Exiting
Datanode
2014-04-17 19:30:35,187 INFO org.apache.hadoop.util.ExitUtil: Exiting with status 0
2014-04-17 19:30:35,189 INFO org.apache.hadoop.hdfs.server.datanode.DataNode:
SHUTDOWN_MSG:
/************************************************************
SHUTDOWN_MSG: Shutting down DataNode at localhost/127.0.0.1
************************************************************/
```

13.4. Inconsistent checkpoint fields

SecondaryNameNode 中的“Inconsistent checkpoint fields”错误原因，可能是因为没有设置好 SecondaryNameNode 上 **core-site.xml** 文件中的“hadoop.**tmp**.dir”。

```
2014-04-17 11:42:18,189 INFO org.apache.hadoop.hdfs.server.namenode.SecondaryNameNode:
Log Size Trigger        :1000000 txns
2014-04-17 11:43:18,365 ERROR
org.apache.hadoop.hdfs.server.namenode.SecondaryNameNode: Exception in doCheckpoint
java.io.IOException: Inconsistent checkpoint fields.
```

```
LV = -56 namespaceID = 1384221685 cTime = 0 ; clusterId =
CID-319b9698-c88d-4fe2-8cb2-c4f440f690d4 ; blockpoolId =
BP-1627258458-172.25.40.171-1397735061985.
Expecting respectively: -56; 476845826; 0; CID-50401d89-a33e-47bf-9d14-914d8f1c4862;
BP-2131387753-172.25.40.171-1397730036484.
        at
org.apache.hadoop.hdfs.server.namenode.CheckpointSignature.validateStorageInfo(CheckpointSignature.java:135)
        at
org.apache.hadoop.hdfs.server.namenode.SecondaryNameNode.doCheckpoint(SecondaryNameNode.java:518)
        at
org.apache.hadoop.hdfs.server.namenode.SecondaryNameNode.doWork(SecondaryNameNode.java:383)
        at
org.apache.hadoop.hdfs.server.namenode.SecondaryNameNode$1.run(SecondaryNameNode.java:349)
        at
org.apache.hadoop.security.SecurityUtil.doAsLoginUserOrFatal(SecurityUtil.java:415)
        at
org.apache.hadoop.hdfs.server.namenode.SecondaryNameNode.run(SecondaryNameNode.java:345)
        at java.lang.Thread.run(Thread.java:744)
```

另外，也请配置好 SecondaryNameNode 上 **hdfs**-site.xml 中的“dfs.datanode.**data**.dir”为合适的值：

```
<property>
  <name>hadoop.tmp.dir</name>
  <value>/data/hadoop/current/tmp</value>
  <description>A base for other temporary directories.</description>
</property>
```

13.5. fs.defaultFS is file:///

在 core-site.xml 中，当只填写了 fs.defaultFS，而 fs.default.name 为默认的 file:///时，会报此错误。解决方法是设置成相同的值。

13.6. a shared edits dir must not be specified if HA is not enabled

该错误可能是因为 hdfs-site.xml 中没有配置 dfs.nameservices 或 dfs.ha.namenodes.mycluster。

13.7. /tmp/dfs/name is in an inconsistent state: storage directory does not exist or is not accessible.

只需按日志中提示的，创建好相应的目录。

13.8. The auxService:mapreduce_shuffle does not exist

问题原因是没有配置 yarn-site.xml 中的“yarn.nodemanager.**aux**-services”，将它的值配置为 **mapreduce_shuffle**，然后重启 yarn 问题即解决。记住所有 yarn 节点都需要修改，包括 ResourceManager 和 NodeManager，如果 NodeManager 上的没有修改，仍然会报这个错误。

13.9. org.apache.hadoop.ipc.Client: Retrying connect to server

该问题，有可能是因为 NodeManager 中的 yarn-site.xml 和 ResourceManager 上的不一致，比如 NodeManager 没有配置 yarn.resourcemanager.ha.rm-ids。

13.10. mapreduce.Job: Running job: job_1445931397013_0001

Hadoop 提交 mapreduce 任务时，卡在 mapreduce.Job: Running job: job_1445931397013_0001 处。
问题原因可能是因为 yarn 的 NodeManager 没起来，可以用 jdk 的 jps 确认下。

该问题也有可能是因为 NodeManager 中的 yarn-site.xml 和 ResourceManager 上的不一致，比如 NodeManager 没有配置 yarn.resourcemanager.ha.rm-ids。

13.11. Could not format one or more JournalNodes

执行“./hdfs namenode -format”时报“Could not format one or more JournalNodes”。
可能是 hdfs-site.xml 中的 dfs.namenode.shared.edits.dir 配置错误，比如重复了，如：

<value>qjournal://hadoop-168-**254**:8485;hadoop-168-**254**:8485;hadoop-168-253:8485;hadoop-168-252:8485;hadoop-168-251:8485/mycluster</value>

修复后，重启 JournalNode，问题可能就解决了。

13.12. org.apache.hadoop.yarn.server.resourcemanager.ResourceManager: Already in standby state

遇到这个错误，可能是 yarn-site.xml 中的 yarn.resourcemanager.webapp.address 配置错误，比如配置成了两个 yarn.resourcemanager.webapp.address.rm1，实际应当是

yarn.resourcemanager.webapp.address.rm1 和 yarn.resourcemanager.webapp.address.rm2。

13.13. No valid image files found

如果是备 NameNode，执行下“hdfs namenode -bootstrapStandby”再启动。

2015-12-01 15:24:39,535 ERROR org.apache.hadoop.hdfs.server.namenode.NameNode: Failed to start namenode.

java.io.FileNotFoundException: **No valid image files found**

at org.apache.hadoop.hdfs.server.namenode.FSImageTransactionalStorageInspector.getLatestImages(FSImageTransactionalStorageInspector.java:165)

at org.apache.hadoop.hdfs.server.namenode.FSImage.loadFSImage(FSImage.java:623)

at org.apache.hadoop.hdfs.server.namenode.FSImage.recoverTransitionRead(FSImage.java:294)

at org.apache.hadoop.hdfs.server.namenode.FSNamesystem.loadFSImage(FSNamesystem.java:975)

at org.apache.hadoop.hdfs.server.namenode.FSNamesystem.loadFromDisk(FSNamesystem.java:681)

at org.apache.hadoop.hdfs.server.namenode.NameNode.loadNamesystem(NameNode.java:584)

at org.apache.hadoop.hdfs.server.namenode.NameNode.initialize(NameNode.java:644)

at org.apache.hadoop.hdfs.server.namenode.NameNode.<init>(NameNode.java:811)

at org.apache.hadoop.hdfs.server.namenode.NameNode.<init>(NameNode.java:795)

at org.apache.hadoop.hdfs.server.namenode.NameNode.createNameNode(NameNode.java:1488)

at org.apache.hadoop.hdfs.server.namenode.NameNode.main(NameNode.java:1554)

2015-12-01 15:24:39,536 INFO org.apache.hadoop.util.ExitUtil: Exiting with status 1

2015-12-01 15:24:39,539 INFO org.apache.hadoop.hdfs.server.namenode.NameNode: SHUTDOWN_MSG:

13.14. xceivercount 4097 exceeds the limit of concurrent xcievers 4096

此错误的原因是 hdfs-site.xml 中的配置项“dfs.datanode.max.xcievers”值 4096 过小，需要改大一点。该错误会导致 hbase 报“notservingregionexception”。

16/04/06 14:30:34 ERROR namenode.NameNode: Failed to start namenode.

13.15. java.lang.IllegalArgumentException: Unable to construct journal, qjournal://hadoop-030:8485;hadoop-031:8454;hadoop-032

执行“hdfs namenode -format”遇到上述错误时，是因为 hdfs-site.xml 中的配置 dfs.namenode.shared.edits.dir 配置错误，其中的 hadoop-032 省了“:8454”部分。

13.16. Bad URI 'qjournal://hadoop-030:8485;hadoop-031:8454;hadoop-032:8454': must identify journal in path component

是因为配置 hdfs-site.xml 中的“dfs.namenode.shared.edits.dir”时，路径少带了 cluster 名。

13.17. 16/04/06 14:48:19 INFO ipc.Client: Retrying connect to server: hadoop-032/10.143.136.211:8454. Already tried 0 time(s); retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)

检查 hdfs-site.xml 中的“dfs.namenode.shared.edits.dir”值，JournalNode 默认端口是 8485，不是 8454，确认是否有写错。JournalNode 端口由 hdfs-site.xml 中的配置项 dfs.journalnode.rpc-address 决定。

13.18. Exception in thread "main" org.apache.hadoop.HadoopIllegalArgumentException: Could not get the namenode ID of this node. You may run zkfc on the node other than namenode.

执行“hdfs zkfc -formatZK”遇到上面这个错误，是因为还没有执行“hdfs namenode -format”。NameNode ID 是在“hdfs namenode -format”时生成的。

13.19. 2016-04-06 17:08:07,690 INFO org.apache.hadoop.hdfs.server.common.Storage: Storage directory [DISK]file:/data3/datanode/data/ has already been used.

以**非 root 用户**启动 DataNode，但启动不了，在它的日志文件中发现如下错误信息：
2016-04-06 17:08:07,707 INFO org.apache.hadoop.hdfs.server.common.Storage: Analyzing storage directories for bpid BP-418073539-10.143.136.207-1459927327462
2016-04-06 17:08:07,707 WARN org.apache.hadoop.hdfs.server.common.Storage: Failed to analyze storage directories for block pool BP-418073539-10.143.136.207-1459927327462
java.io.IOException: BlockPoolSliceStorage.recoverTransitionRead: attempt to load an used block storage: /data3/datanode/data/current/BP-418073539-10.143.136.207-1459927327462
继续寻找，会发现还存在如何错误提示：

Invalid dfs.datanode.data.dir /data3/datanode/data:
EPERM: **Operation not permitted**
使用命令“ls -l”检查目录/data3/datanode/data 的权限设置，发现 owner 为 root，原因是因为之前使用 root 启动过 DataNode，将 owner 改过来即可解决此问题。

13.20. 2016-04-06 18:00:26,939 WARN org.apache.hadoop.hdfs.server.datanode.DataNode: Problem connecting to server: hadoop-031/10.143.136.208:8020

DataNode 的日志文件不停地记录如下日志，是因为 DataNode 将作为主 NameNode，但实际上 10.143.136.208 并没有启动，主 NameNode 不是它。这个并不表示 DataNode 没有起来，而是因为 DataNode 会同时和主 NameNode 和备 NameNode 建立心跳，当备 NameNode 没有起来时，有这些日志是正常现象。
2016-04-06 18:00:32,940 INFO org.apache.hadoop.ipc.Client: Retrying connect to server: hadoop-031/10.143.136.208:**8020**. Already tried 0 time(s); retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)
2016-04-06 17:55:44,555 INFO org.apache.hadoop.hdfs.server.datanode.DataNode: Namenode Block pool BP-418073539-10.143.136.207-1459927327462 (Datanode Uuid 2d115d45-fd48-4e86-97b1-e74a1f87e1ca) service to hadoop-030/10.143.136.207:8020 **trying to claim ACTIVE state** with txid=1
“**trying to claim ACTIVE state**”出自于 hadoop/hdfs/server/datanode/BPOfferService.java 中的 updateActorStatesFromHeartbeat()。

2016-04-06 17:55:49,893 INFO org.apache.hadoop.ipc.Client: **Retrying connect to server**: hadoop-031/10.143.136.208:8020. Already tried 5 time(s); retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)
“**Retrying connect to server**”出自于 hadoop/ipc/Client.java 中的 handleConnectionTimeout() 和 handleConnectionFailure()。

14. 相关文档

《HBase-0.98.0 分布式安装指南》
《Hive 0.12.0 安装指南》
《ZooKeeper-3.4.6 分布式安装指南》
《Hadoop 2.3.0 源码反向工程》
《在 Linux 上编译 Hadoop-2.7.2》
《Accumulo-1.5.1 安装指南》
《Drill 1.0.0 安装指南》
《Shark 0.9.1 安装指南》