# Hadoop 安装

## 1 重新编译

一般hadoop需要在自己的linux环境下重新将源代码编译一下，为什么hadoop要自己再次编译一下?

看下hadoop官网给的相关说明（翻译后的）官网链接 http://hadoop.apache.org/docs/r1.0.4/cn/native_libraries.html

`Hadoop`是使用`Java`语言开发的，但是有一些需求和操作并不适合使用`Java`,所以就引入了本地库`Native Libraries`的概念。说白了，就是`Hadoop`的某些功能，必须通过`JNT`来协调`Java`类文件和`Native`代码生成的库文件一起才能工作。`linux`系统要运行`Native`代码，首先要将`Native`编译成目标`CPU`架构的[.so]文件。而不同的处理器架构，需要编译出相应平台的动态库[.so] 文件，才能被正确的执行，所以最好重新编译一次`hadoop`源码，让[.so]文件与自己处理器相对应。

### 1.1 准备

Apach官方下载列表：http://archive.apache.org/dist/

1. 下载`hadoop-3.2.1-src.tar.gz` 源码包，先查看`Hadoop`源码包中的`BUILDING.txt`文本里面说明了各个系统平台下的环境需求准备.

```
Unix System
* JDK 1.8 （必须）
* Maven 3.3 or later （必须）
* ProtocolBuffer 2.5.0 （必须且必须是这个版本）
* CMake 3.1 or newer (if compiling native code) （必须，yum安装或者下载tar包解压都行，后面有具体操作）
* Zlib devel (if compiling native code) （本人在实际操作的时候是必须要装的，但是在低版本的Hadoop貌似是不需要）
* Cyrus SASL devel (if compiling native code)
* One of the compilers that support thread_local storage: GCC 4.8.1 or later, Visual Studio,
Clang (community version), Clang (version for iOS 9 and later) (if compiling native code)
* openssl devel (if compiling native hadoop-pipes and to get the best HDFS encryption performance)（需要，直接使用yum源安装即可）
* Linux FUSE (Filesystem in Userspace) version 2.6 or above (if compiling fuse_dfs)（以下都不需要管）
* Doxygen ( if compiling libhdfspp and generating the documents )
* Internet connection for first build (to fetch all Maven and Hadoop dependencies)
* python (for releasedocs)
* bats (for shell code testing)
* Node.js / bower / Ember-cli (for YARN UI v2 building)
```

2. jdk-8u261-linux-x64.tar.gz
3. `apache-maven-3.6.3-bin.tar.gz`:https://mirrors.tuna.tsinghua.edu.cn/apache/maven/maven-3/3.6.3/binaries/
4. `apache-ant-1.9.15-bin.tar.gz`:https://mirrors.tuna.tsinghua.edu.cn/apache/ant/binaries/(软件编译、测试、部署等步骤联

系在一起加以自动化的一个工具)

5. `cmake-3.18.4-Linux-x86_64.tar.gz`:https://github.com/Kitware/CMake/releases/download/v3.18.4/cmake-3.18.4-Linux-x86_64.tar.gz
6. `v1.2.11.tar.gz(Zlib)`:https://github.com/madler/zlib/archive/v1.2.11.tar.gz
7. `protobuf-2.5.0.tar.gz`:https://github.com/protocolbuffers/protobuf/releases/download/v2.5.0/protobuf-2.5.0.tar.gz
8. snappy-1.1.3.tar.gz:https://src.fedoraproject.org/repo/pkgs/snappy/

## 1.2 安装环境

1. `JDK`安装

```
# 解压tar包到指定目录
mkdir /usr/local/java
tar -zxvf jdk-8u261-linux-x64.tar.gz -C
/usr/local/java # 大写-C
# 配置环境变量
vim /etc/profile
export JAVA_HOME=/usr/local/java/jdk1.8.0_261
export
CLASSPATH=.:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools
.jar
export PATH=$JAVA_HOME/bin:$PATH
# 立即生效
source /etc/profile
# 验证
java -version
```

2. `maven`安装

```
# 解压tar包到指定目录
tar -zxvf apache-maven-3.6.3-bin.tar.gz -C /usr/local
# 配置环境变量
vim /etc/profile
export MAVEN_HOME=/usr/local/apache-maven-3.6.3
export PATH=$MAVEN_HOME/bin:$PATH
# 立即生效
source /etc/profile
# 输入mvn -v 验证是否安装成功
mvn -v
```

3. `ant`安装

```
# 解压tar包到指定目录
tar -zxvf apache-ant-1.9.15-bin.tar.gz -C /usr/local
# 配置环境变量
vim /etc/profile
export ANT_HOME=/usr/local/apache-ant-1.9.15
export PATH=$ANT_HOME/bin:$PATH
# 立即生效
source /etc/profile
# 输入ant -version 验证是否安装成功
ant -version
```

4. `cmake`安装

```
# 安装cmake之前需要先安装cmake的依赖环境
yum install -y gcc gcc-c++
# 解压tar包到指定目录
tar -zxvf cmake-3.18.4-Linux-x86_64.tar.gz -C
/usr/local
# 由于下载的版本是已经编译好的，值需要添加环境变量即可
# 然后还是最简单的配置环境变量
vim /etc/profile
export CMAKE_HOME=/usr/local/cmake-3.18.4-Linux-x86_64
export PATH=$CMAKE_HOME/bin:$PATH
# 立即生效
source /etc/profile
# 输入cmake -version验证是否成功
cmake -version
```

5. `zlib`安装

```
# 解压tar包
tar -zxvf zlib-1.2.11.tar.gz
# 进入zlib目录进行编译
cd zlib-1.2.11
# - -prefix 指定安装位置
./configure --prefix=/usr/local/zlib
make && make install
# 写入动态库的路径
echo "/usr/local/zlib/lib" >> /etc/ld.so.conf
ldconfig -v
```

6. `protobuf`安装

```
# 解压tar包到指定目录
tar -zxvf protobuf-2.5.0.tar.gz
# 进入protobuf目录执行一下命令
cd protobuf-2.5.0
./configure --prefix=/usr/local/protobuf
make && make install
# 全部执行完并且没有错误之后配置环境变量
vim /etc/profile
export PROTOC_HOME=/usr/local/protobuf
export PATH=$PROTOC_HOME/bin:$PATH
# 输入 protoc --version 验证是否安装成功
source /etc/profile
protoc --version
```

7. `ncurses-devel`库

```
yum install ncurses-devel -y
```

8. 重装`openssl`并配置`lib`路径

```
# 下载
wget https://www.openssl.org/source/openssl-
1.1.1g.tar.gz
# 解压tar
tar -zxvf openssl-1.1.1g.tar.gz
# 进入目录
cd openssl-1.1.1g
# 指定安装目录
./config --prefix=/usr/local/openssl
# 测试
make test
# 安装
make install
```

```
# 创建软链接
ln -s /usr/local/openssl/include/openssl
/usr/include/openssl
ln -s /usr/local/openssl/lib/libssl.so
/usr/lib64/libssl.so
echo "/usr/local/openssl/lib" >> /etc/ld.so.conf
ldconfig -v
# 查看版本
openssl version
```

9. 安装`libssl`

```
yum install -y  libssl*
```

10. 安装`snappy`

```
# download snappy
http://pkgs.fedoraproject.org/repo/pkgs/snappy/

# install snappy
tar xvfz snappy-1.1.3.tar.gz
cd snappy-1.1.3
./configure --prefix=/usr/local/snappy
make
make install

# download snzip
https://bintray.com/kubo/generic/snzip

# install snzip
tar xvfz snzip-1.0.4.tar.gz
cd snzip-1.0.4
./configure --with-snappy=/usr/local/snappy
make
```

```
make install

./configure --with-default-format=snzip
```

## 1.3 编译

1. 设置`Maven`镜像

```
<!-- 阿里的maven镜像仓库 -->
<mirror>
  <id>nexus-aliyun</id>
  <mirrorOf>central</mirrorOf>
  <name>Nexus aliyun</name>

<url>http://maven.aliyun.com/nexus/content/groups/public<
/url>
</mirror>
```

2. 开始编译

```
# 将Hadoop的源码包解压
tar -zxvf hadoop-3.2.0-src.tar.gz
# 进入到Hadoop源码包
cd hadoop-3.2.0-src
# 执行命令，会等较长时间
# 如果中途等待时间再长 可以打断，继续执行
mvn clean
mvn package -Pdist,native,docs  -DskipTests -Dtar -
Drequire.snappy -e -X
# 经过漫长的等待看到一堆SUCCESS以及BUILD SUCCESS并且没有报错
就是成功了，然后就可以开始快乐的Hadoop之旅了
```

3. 最后编译成功后的`Hadoop`包在源码包的`hadoop-3.2.1-src/hadoop-dist/target`目录下。

# 2 Hadoop 安装

## 2.1 配置文件

需要配置的文件如下：

- /etc/hadoop/hadoop-env.sh
- /etc/hadoop/core-site.xml
- /etc/hadoop/hdfs-site.xml
- /etc/hadoop/mapred-site.xml
- /etc/hadoop/yarn-site.xml
- /etc/hadoop/workers

### hadoop-env.sh

在文件中找到以下内容，解开注释，添加`JAVA_HOME`路径：

```
# The java implementation to use. By default, this
environment
# variable is REQUIRED on ALL platforms except OS X!
export JAVA_HOME=/usr/local/java/jdk1.8.0_261
```

### core-site.xml

| 参数 | 属性值 | 解释 |
| --- | --- | --- |
| fs.defaultFS | NameNode URI | hdfs://host:port/ |
| io.file.buffer.size | 131072 | SequenceFiles文件中.读写缓存size设定 |

示例:

```
<configuration>
    <property>
        <name>fs.defaultFS</name>
        <value>hdfs://node-1:900</value>
        <description>192.168.1.100为服务器IP地址，其实也可以使用主机名</description>
    </property>
    <property>
        <name>hadoop.tmp.dir</name>
        <value>/var/hadoop/tmp</value>
      <description>指定 namenode 上本地的 hadoop 临时文件夹</description>
    </property>
    <property>
        <name>io.file.buffer.size</name>
        <value>131072</value>
        <description>该属性值单位为KB，131072KB即为默认的64M</description>
    </property>
</configuration>
```

### hdfs-site.xml

配置`NameNode`

| 参数 | 属性值 | 解释 |
| --- | --- | --- |
| dfs.namenode.name.dir | 在本地文件系统所在的NameNode的存储空间和持续化处理日志 | 如果这是一个以逗号分隔的目录列表，然 后将名称表被复制的所有目录，以备不时 需。 |
| dfs.namenode.hosts/ dfs.namenode.hosts.exclude | Datanodes permitted/excluded 列表 | 如有必要，可以使用这些文件来控制允许 数据节点的列表 |
| dfs.blocksize | 268435456 | 大型的文件系统HDFS块大小为256MB |
| dfs.namenode.handler.count | 100 | 设置更多的namenode线程，处理从datanode发出的大量RPC请求 |

伪分布式只需要配置

```
<configuration>
   <property>
       <name>dfs.replication</name>
       <value>1</value>
      <description>分片数量，伪分布式将其配置成1即可
</description>
   </property>
</configuration>
```

完全分布式配置

```
<configuration>
    <property>
        <name>dfs.replication</name>
        <value>2</value>
```

```
        <description>分片数量，伪分布式将其配置成1即可
</description>
    </property>
    <property>
        <name>dfs.namenode.name.dir</name>
        <value>file://${hadoop.tmp.dir}/dfs/data</value>
        <description>命名空间和事务在本地文件系统永久存储的路
径</description>
    </property>
    <property>
        <name>dfs.namenode.hosts</name>
        <value>datanode1, datanode2</value>
        <description>datanode1, datanode2分别对应DataNode
所在服务器主机名</description>
    </property>
    <property>
        <name>dfs.blocksize</name>
        <value>268435456</value>
        <description>大文件系统HDFS块大小为256M，默认值为
64M</description>
    </property>
    <property>
        <name>dfs.namenode.handler.count</name>
        <value>100</value>
        <description>更多的NameNode服务器线程处理来自
DataNodes的RPCS</description>
    </property>
</configuration>
```

### mapred-site.xml

配置`mapreduce`

| 参数 | 属性值 | 解释 |
| --- | --- | --- |
| mapreduce.framework.name | yarn | 执行框架设置为Hadoop YARN. |
| mapreduce.map.memory.mb | 1536 | 对maps更大的资源限制的. |
| mapreduce.map.java.opts | -Xmx2014M | maps中对jvm child设置更大的堆大小 |
| mapreduce.reduce.memory.mb | 3072 | 设置 reduces对于较大的资源限制 |
| mapreduce.reduce.java.opts | -Xmx2560M | reduces对 jvm child设置更大的堆大小 |
| mapreduce.task.io.sort.mb | 512 | 更高的内存限制，而对数据进行排序的效率 |
| mapreduce.task.io.sort.factor | 100 | 在文件排序中更多的流合并为一次 |
| mapreduce.reduce.shuffle.parallelcopies | 50 | 通过reduces从很多的map中读取较多的平行副本 |

## 说明配置

```
<configuration>
    <property>
        <name> mapreduce.framework.name</name>
        <value>yarn</value>
        <description>执行框架设置为Hadoop
YARN</description>
    </property>
    <property>
        <name>mapreduce.map.memory.mb</name>
        <value>1536</value>
        <description>对maps更大的资源限制的</description>
    </property>
```

```
    <property>
        <name>mapreduce.map.java.opts</name>
        <value>-Xmx2014M</value>
        <description>maps中对jvm child设置更大的堆大
小</description>
    </property>
    <property>
        <name>mapreduce.reduce.memory.mb</name>
        <value>3072</value>
        <description>设置 reduces对于较大的资源限
制</description>
    </property>
    <property>
        <name>mapreduce.reduce.java.opts</name>
        <value>-Xmx2560M</value>
        <description>reduces对 jvm child设置更大的堆大
小</description>
    </property>
    <property>
        <name>mapreduce.task.io.sort</name>
        <value>512</value>
        <description>更高的内存限制，而对数据进行排序的效率
</description>
    </property>
    <property>
        <name>mapreduce.task.io.sort.factor</name>
        <value>100</value>
        <description>在文件排序中更多的流合并为一次
</description>
    </property>
    <property>

 <name>mapreduce.reduce.shuffle.parallelcopies</name>
```

```
        <value>50</value>
        <description>通过reduces从很多的map中读取较多的平行副本</description>
    </property>
</configuration>
```

伪分布式和完全分布式配置

```
<configuration>
    <property>
        <name>mapreduce.framework.name</name>
        <value>yarn</value>
       <description>执行框架设置为Hadoop YARN</description>
    </property>
</configuration>
```

### yarn-site.xml

配置`ResourceManager` 和`NodeManager`

1. 配置`ResourceManager` 如下

| 参数 | 属性值 | 解释 |
| --- | --- | --- |
| yarn.resourcemanager.address | 客户端对ResourceManager主机通过 host:port 提交作业 | host:port |
| yarn.resourcemanager.scheduler.address | ApplicationMasters 通过ResourceManager主机访问host:port跟踪调度程序获资源 | host:port |
| yarn.resourcemanager.resource-tracker.address | NodeManagers通过ResourceManager主机访问host:port | host:port |
| yarn.resourcemanager.admin.address | 管理命令通过ResourceManager主机访问host:port | host:port |
| yarn.resourcemanager.webapp.address | ResourceManager web页面host:port. | host:port |
| yarn.resourcemanager.scheduler.class | ResourceManager 调度类（Scheduler class） | CapacityScheduler（推荐），FairScheduler（也推荐），orFifoScheduler |
| yarn.scheduler.minimum-allocation-mb | 每个容器内存最低限额分配到的资源管理器要求 | 以MB为单位 |
| yarn.scheduler.maximum-allocation-mb | 资源管理器分配给每个容器的内存最大限制 | 以MB为单位 |
| yarn.resourcemanager.nodes.include-path/ yarn.resourcemanager.nodes.exclude-path | NodeManagers的permitted/excluded列表 | 如有必要，可使用这些文件来控制允许NodeManagers列表 |

示例：

```
<configuration>
    <property>
        <name>yarn.resourcemanager.address</name>
        <value>192.168.1.100:8081</value>
        <description>IP地址192.168.1.100也可替换为主机
名</description>
    </property>
    <property>

 <name>yarn.resourcemanager.scheduler.address</name>
        <value>192.168.1.100:8082</value>
        <description>IP地址192.168.1.100也可替换为主机
名</description>
```

```
    </property>
    <property>
        <name>yarn.resourcemanager.resource-
tracker.address</name>
        <value>192.168.1.100:8083</value>
        <description>IP地址192.168.1.100也可替换为主机
名</description>
    </property>
    <property>
        <name>yarn.resourcemanager.admin.address</name>
        <value>192.168.1.100:8084</value>
        <description>IP地址192.168.1.100也可替换为主机
名</description>
    </property>
    <property>
        <name>yarn.resourcemanager.webapp.address</name>
        <value>192.168.1.100:8085</value>
        <description>IP地址192.168.1.100也可替换为主机
名</description>
    </property>
    <property>
        <name>yarn.resourcemanager.scheduler.class</name>
        <value>FairScheduler</value>
        <description>常用类：CapacityScheduler、
FairScheduler、orFifoScheduler</description>
    </property>
    <property>
        <name>yarn.scheduler.minimum</name>
        <value>100</value>
        <description>单位：MB</description>
    </property>
    <property>
        <name>yarn.scheduler.maximum</name>
```

```
        <value>256</value>
        <description>单位：MB</description>
    </property>
    <property>
        <name>yarn.resourcemanager.nodes.include-
path</name>
        <value>nodeManager1, nodeManager2</value>
        <description>nodeManager1, nodeManager2分别对应服
务器主机名</description>
    </property>
</configuration>
```

## 2. 配置`NodeManager`如下

| 参数 | 属性值 | 解释 |
| --- | --- | --- |
| yarn.nodemanager.resource.memory-mb | givenNodeManager即资源的可用物理内存，以MB为单位 | 定义在节点管理器总的可用资源，以提供给运行容器 |
| yarn.nodemanager.vmem-pmem-ratio | 最大比率为一些任务的虚拟内存使用量可能会超过物理内存率 | 每个任务的虚拟内存的使用可以通过这个比例超过了物理内存的限制。虚拟内存的使用上的节点管理器任务的总量可以通过这个比率超过其物理内存的使用 |
| yarn.nodemanager.local-dirs | 数据写入本地文件系统路径的列表用逗号分隔 | 多条存储路径可以提高磁盘的读写速度 |
| yarn.nodemanager.log-dirs | 本地文件系统日志路径的列表逗号分隔 | 多条存储路径可以提高磁盘的读写速度 |
| yarn.nodemanager.log.retain-seconds | 10800 | 如果日志聚合被禁用。默认的时间（以秒为单位）保留在节点管理器只适用日志文件 |
| yarn.nodemanager.remote-app-log-dir | logs | HDFS目录下的应用程序日志移动应用上完成。需要设置相应的权限。仅适用日志聚合功能 |
| yarn.nodemanager.remote-app-log-dir-suffix | logs | 后缀追加到远程日志目录。日志将被汇总到`yarn.nodemanager.remoteapp-logdir/{user}/${thisParam}`仅适用日志聚合功能。 |
| yarn.nodemanager.aux-services | mapreduce-shuffle | Shuffle service 需要加以设置的Map Reduce的应用程序服务 |

## 示例

```
<configuration>
    <property>
        <name>yarn.nodemanager.resource.memory-mb</name>
        <value>256</value>
        <description>单位为MB</description>
    </property>
    <property>
        <name>yarn.nodemanager.vmem-pmem-ratio</name>
        <value>90</value>
        <description>百分比</description>
    </property>
    <property>
        <name>yarn.nodemanager.local-dirs</name>
        <value>/usr/local/hadoop/tmp/nodemanager</value>
        <description>列表用逗号分隔</description>
    </property>
    <property>
        <name>yarn.nodemanager.log-dirs</name>

 <value>/usr/local/hadoop/tmp/nodemanager/logs</value>
        <description>列表用逗号分隔</description>
    </property>
    <property>
        <name>yarn.nodemanager.log.retain-seconds</name>
        <value>10800</value>
        <description>单位为S</description>
    </property>
    <property>
        <name>yarn.nodemanager.aux-services</name>
        <value>mapreduce-shuffle</value>
        <description>Shuffle service 需要加以设置的
MapReduce的应用程序服务</description>
    </property>
```

```
</configuration>
```

伪分布式配置和完全分布式配置

```
<configuration>
    <property>
        <name>yarn.resourcemanager.hostname</name>
        <value>node-1</value>
        <description>IP地址192.168.1.100也可替换为主机
名</description>
    </property>

    <property>
        <name>yarn.nodemanager.aux-services</name>
        <value>mapreduce_shuffle</value>
       <description>Shuffle service 需要加以设置的
MapReduce的应用程序服务 nodemanager 运行的附属服务，要配置成
mapreduce_shuffle，才可以运行 MapReduce 程序默认
值</description>
    </property>
</configuration>
```

**workers**

添加从节点。伪分布式模式中，只有一个 node-1 节点：

```
node-1
```

Hadoop 3.0 以前的版本是 slaves 文件

# 2.3 完全分布式集群搭建

Hadoop-3.2.1完全分布模式配置步骤如下(CentOS 7.x)：

1. 固定集群中每个节点的IP、机器名、IP与机器名的映射；
2. 安装和配置JDK。
3. 配置SSH无密登录。
4. 在单台机器上安装并配置Hadoop。
5. 将配置好的Hadoop远程拷贝到集群其他节点上。
6. 格式化HDFS文件系统（只格式化一次）。
7. 启动Hadoop集群。
8. 在Hadoop上执行MR程序。

**本次高可用集群搭建采用了三台服务器，关系如下表**

| 主机名称 | namenode | datanode | zookeeper | hbase | hive |
| --- | --- | --- | --- | --- | --- |
| hadoop-master | √ | √ | √ | √ | √ |
| hadoop-worker1 | | √ | √ | √ | |
| hadoop-worker2 | | √ | √ | √ | |

## 2.3.1 集群准备

安装好JDK并配置好环境变量(也包含hadoop的环境变量，虽然hadoop没有安装)的centos，ip地址最好设置为静态，

创建相关目录

```
mkdir -p  /opt/hadoop/tmp /opt/hadoop/hdfs
/opt/hadoop/hdfs/data  /opt/hadoop/hdfs/name
```

最后复制三份

## 2.3.2 设置主机名称

```
hostnamectl set-hostname 主机名
#主节点的主机名设置为hadoop-master，两个从节点的主机名设置为hadoop-worker1以及hadoop-worker2
# eg:hostnamectl set-hostname hadoop-master
# eg:hostnamectl set-hostname hadoop-worker1
# eg:hostnamectl set-hostname hadoop-worker2
```

### 2.3.3 修改hosts文件添加ip映射关系

```
# 使用如下命令修改hosts文件
vim /etc/hosts
# 节点ip  主机名
# eg: 192.168.21.114 hadoop-master
# eg: 192.168.21.115 hadoop-worker1
# eg: 192.168.21.116 hadoop-worker2
```

### 2.3.4 设置集群ssh免密登录

namenode作为客户端，要实现无密码公钥认证，连接到服务端datanode上时，需要在namenode上生成一个密钥对，包括一个公钥和一个私钥，而后将公钥复制到datanode上。当namenode通过ssh连接datanode时，datanode就会生成一个随机数并用namenode的公钥对随机数进行加密，并发送给namenode。namenode 收到加密数之后再用私钥进行解密，并将解密数回传给datanode，datanode确认解密数无误之后就允许namenode进行连接了。这就是一个公钥认证过程，其间不需要用户手工输入密码。重要过程是将namenode公钥复制到datanode上。

- 使用命令 `ssh-keygen -t rsa` 生成 ssh 密钥对，均使用默认选项即可，在 `~/.ssh` 隐藏目录下生成 `id_rsa`（私钥）和 `id_rsa.pub`（公钥）

- 将公钥复制到需要免密登录的机器上。例如：执行 `ssh-copy-id 192.168.220.11` 或 `ssh-copy-id node-2`

1. 在每台机器上，执行如下命令,可以修改显示的名称

```
cd ~
ssh localhost
ssh exit                    # 记得最后通过这个命令退出ssh连接
```

2. 在namenode(master主机)上，使用如下命令生成公私钥：

```
cd .ssh
ssh-keygen -t rsa
```

然后一路回车，在.ssh目录下生成公私钥

3. 将namenode(master主机)上的公钥分别加入hadoop-master、hadoop-worker1和hadoop-worker2机器的授权文件中。 在namenode(master主机)上，执行如下命令：

```
ssh-copy-id hadoop-master
ssh-copy-id hadoop-worker1
ssh-copy-id hadoop-worker2
```

4. 测试。在master节点机器上，使用ssh分别连接master、worker1和worker2:

```
ssh hadoop-master
exit

ssh hadoop-worker1
exit
```

## 2.3.5 在master节点上安装Hadoop

```
# 将Hadoop的源码包解压
tar -zxvf hadoop-3.2.1.tar.gz
# 复制到 /usr/local目录下
cp -rp hadoop-3.2.1 /usr/local
# 修改文件夹的名称
mv hadoop-3.2.1 hadoop
# 由于之前已经进行了环境变量的配置
# 输入hadoop version验证是否成功
hadoop version
```

## 2.3.6 在master节点上配置Hadoop

配置Hadoop，共需要配置6个文件，均位于Hadoop安装目录下的"etc/hadoop/"子目录下。首先进入到该目录下。

1. 配置hadoop-env.sh文件,找到并修改JAVA_HOME属性的值

```
# The java implementation to use. By default, this
environment
# variable is REQUIRED on ALL platforms except OS X!
export JAVA_HOME=/usr/local/java/jdk1.8.0_261
```

2. 配置core-site.xml文件

```
<configuration>
  <!-- 192.168.21.114为服务器IP地址，其实也可以使用主机名 --
>
    <property>
        <name>fs.defaultFS</name>
        <value>hdfs://hadoop-master:9000</value>
```

```
        <description>192.168.21.114为服务器IP地址，其实也可以使用主机名</description>
    </property>
  <!-- 指定 namenode 上本地的 hadoop 临时文件夹 -->
    <property>
        <name>hadoop.tmp.dir</name>
        <value>/opt/hadoop/tmp</value>
      <description>指定 namenode 上本地的 hadoop 临时文件夹</description>
    </property>
  <!-- 该属性值单位为KB，131072KB即为默认的64M -->
    <property>
        <name>io.file.buffer.size</name>
        <value>131072</value>
        <description></description>
    </property>
</configuration>
```

3. 配置hdfs-site.xml文件

```
<configuration>
<!-- hdfs的副本数设置。也就是上传一个文件，其分割为block块后，每个block的冗余副本个数，一般跟datanode节点数相等 -->
    <property>
        <name>dfs.replication</name>
        <value>3</value>
    </property>
    <!--dfs namenode web界面的监听地址-->
    <property>
        <name>dfs.namenode.http-address</name>
        <value>hadoop-master:9870</value>
    </property>
```

```
    <!-- dfs保存namenode数据的路径-->
    <property>
        <name>dfs.namenode.name.dir</name>
        <value>/opt/hadoop/hdfs/name</value>
    </property>

    <!-- dfs保存datanode数据的路径-->
    <property>
        <name>dfs.datanode.data.dir</name>
        <value>/opt/hadoop/hdfs/data</value>
    </property>
</configuration>
```

4. 修改mapred-site.xml配置文件

```
<configuration>
<!-- 指定mr框架为yarn方式,Hadoop二代MP也基于资源管理系统Yarn来运行-->
    <property>
        <name>mapreduce.framework.name</name>
        <value>yarn</value>
    </property>
</configuration>
```

5. 修改yarn-site.xml

```
<configuration>
<!-- 这里面主要指定了一些节点资源管理器nodemanager，以及总资源管理器resourcemanager的配置-->
<!-- yarn是为了支持mapreduce这个模型-->
  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
  </property>
```

```
  <property>
    <name>yarn.nodemanager.env-whitelist</name>
 <value>JAVA_HOME,HADOOP_COMMON_HOME,HADOOP_HDFS_HOME,HAD
OOP_CONF_DIR,CLASSPATH_PREPEND_DISTCACHE,HADOOP_YARN_HOME
,HADOOP_MAPRED_HOME</value>
  </property>
  <!-- yarn总管理的主机-->
  <property>
    <name>yarn.resourcemanager.hostname</name>
    <value>hadoop-master</value>
  </property>
</configuration>
```

6. 配置从节点workers

将里面的localhost删除，写入集群中从节点的机器名，每个一行，然后保存文件

```
hadoop-master
hadoop-worker1
hadoop-worker2
```

## 2.3.7 复制hadoop

将配置好的hadoop文件夹，分别拷贝到worker1和worker2的/usr/local目录下

```
tar -cf hadoop.tar.gz hadoop
scp hadoop.tar.gz root@hadoop-worker1:/usr/local
scp hadoop.tar.gz root@hadoop-worker2:/usr/local

tar -xf hadoop.tar.gz
```

### 2.3.8 启动hdfs

如果因为某些原因需要从头重新配置集群，那么在重新格式化HDFS之前，先把每个节点中Hadoop下的dfs目录删除。这个目录是在hdfs-site.xml文件中自己指定的，其下有两个子目录name和data，重新格式化之前必须删除它们。

```
# 格式化文件系统 只能第一次使用
hdfs namenode -format
# 启动nodename
hdfs --daemon start namenode
hdfs --daemon start datanode
jps
```

hadoop-worker1，hadoop-worker2启动datanode

```
hdfs --daemon start datanode
jps
```

hdfs web界面： http://hadoop-master:9870

### 2.3.9 启动yarn

hadoop-master启动resourcemanager、nodemanager

```
yarn --daemon start resourcemanager
yarn --daemon start nodemsnager
```

hadoop-worker1、hadoop-worker2启动nodemanager

```
yarn --daemon start nodemanager
```

yarn web界面： http://hadoop-master:8088/cluster/nodes

http://192.168.21.114:9870 （nameNode information 存在于主节点上 ）

http://192.168.21.114:8088/cluster （All Applications 集群运行状态，以及任务执行服务）

http://192.168.21.(115|116):8042/node （NodeManager 数据节点管理服务，存在于DataNode数据节点上）

http://192.168.21.(115|116):9864/datanode.html （DataNode，数据存储， 存在于DataNode数据节点上）

# 3 Hive 安装

本文的`hive`、`MySQL`均只安装在`Master`节点上，实际生产环境中，需根据实际情况调整

下载地址：https://www.apache.org/dyn/closer.lua/hbase/2.2.6/hbase-2.2.6-bin.tar.gz

1. 安装

```
tar -zxvf apache-hive-3.1.2-bin.tar.gz -C /usr/local   # 解压到/usr/local中
cd /usr/local/
mv apache-hive-3.1.2-bin hive        # 将文件夹名改为hive
```

2. 将 /usr/local/hadoop/share/hadoop/common/lib/guava-27.0-jre.jar 拷贝并替换掉 /usr/local/hive/lib/guava-19.0.jar

```
cp /usr/local/hadoop/share/hadoop/common/lib/guava-27.0-jre.jar /usr/local/hive/lib
rm -rf guava-19.0.jar
```

3. 配置环境变量

```
vim /etc/profile
export HIVE_HOME=/usr/local/hive
export PATH=$HIVE_HOME/bin:$PATH
# 立即生效
source /etc/profile
hive --version
```

3. 在`/usr/local/hive/conf`下进行如下操作

```
cd /usr/local/hive/conf
cp hive-env.sh.template hive-env.sh
# 编辑 hive-env.sh 添加
export HADOOP_HOME=/usr/local/hadoop
# 创建 hive-site.xml 并添加配置
vim hive-site.xml
```

添加如下配置

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?><!--
   Licensed to the Apache Software Foundation (ASF) under one or more
   contributor license agreements.  See the NOTICE file distributed with
   this work for additional information regarding copyright ownership.
```

```
   The ASF licenses this file to You under the Apache
License, Version 2.0
   (the "License"); you may not use this file except in
compliance with
   the License.  You may obtain a copy of the License at

       http://www.apache.org/licenses/LICENSE-2.0

   Unless required by applicable law or agreed to in
writing, software
   distributed under the License is distributed on an "AS
IS" BASIS,
   WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
express or implied.
   See the License for the specific language governing
permissions and
   limitations under the License.
--><configuration>
  <!-- WARNING!!! This file is auto generated for
documentation purposes ONLY! -->
  <!-- WARNING!!! Any changes you make to this file will
be ignored by Hive.   -->
  <!-- WARNING!!! You must make your changes in hive-
site.xml instead.         -->
  <!-- Hive Execution Parameters -->
<!-- 插入一下代码-->
    <property>
        <!--用户名 -->
        <name>javax.jdo.option.ConnectionUserName</name>
        <value>root</value>
    </property>
    <property>
        <!--密码-->
```

```
        <name>javax.jdo.option.ConnectionPassword</name>
        <value>123456</value>
    </property>
   <property>
        <name>javax.jdo.option.ConnectionURL</name>
        <value>jdbc:mysql://localhost:3306/hive?
useSSL=false&amp;rewriteBatchedStatements=true&amp;useSer
verPrepStmts=true&amp;cachePrepStmts=true&amp;autoReconne
ct=true&amp;failOverReadOnly=false</value>
    </property>
    <property>
        <!--mysql驱动程序-->

 <name>javax.jdo.option.ConnectionDriverName</name>
        <value>com.mysql.cj.jdbc.Driver</value>
    </property>
    <property>
        <name>hive.metastore.schema.verification</name>
        <value>false</value>
    </property>
<!-- 到此结束代码 -->
</configuration>
```

5. 安装并配置mysql

拷贝`MySQL`驱动包到`/usr/local/hive/lib`

```
# 解包:
rpm2cpio mysql-connector-java-8.0.18-1.el7.noarch.rpm |
cpio -div
# 拷贝
cp usr/share/java/mysql-connector-java.jar
/usr/local/hive/lib
```

安装MySQL

```
# 1. 下载MySQL安装包时，要注意*-el7-*中的数字7对应操作系统的版本，必须保证版本对应
# 2.https://dev.mysql.com/doc/refman/8.0/en/linux-installation-rpm.html 参考官方安装步骤
# 3.解压 tar -xf mysql-8.0.18-1.el7.x86_64.rpm-bundle.tar
# 4.可以将不需要的包进行删除
# 5. 安装
yum install mysql-community-{server,client,common,libs}-* -y
# 6.启动守护进程
systemctl start mysqld
# 7.查看状态
systemctl status mysqld
# 8.获取默认密码
grep 'temporary password' /var/log/mysqld.log
# 9 .登陆 mysql 修改密码，密码要求严格，继续10步操作
# 10 SHOW VARIABLES LIKE 'validate_password.%' # 先看看当前的密码验证策略
# 策略说明 validate_password.length 是密码的最小长度，默认是8，我们把它改成6
# 输入：set global validate_password.length=6;
# validate_password.policy 验证密码的复杂程度，我们把它改成0
# 输入：set global validate_password.policy=0;
# validate_password.check_user_name 用户名检查，用户名和密码不能相同，我们也把它关掉
# 输入：set global validate_password.check_user_name=off;
# 再执行修改密码的命令
# 输入：ALTER USER 'root'@'localhost' IDENTIFIED BY '123456';
# 使用mysql数据库
use mysql;
```

```
# 设置权限（为root分配权限，以便可以远程连接）
GRANT ALL ON *.* TO 'root'@'%' #如果报错 进行查询 select user,host from user;
# 解决方法 update user set host = '%' where user = 'root' and host='localhost' 或 update user set host='%' where user='root';
# 由于Mysql5.6以上的版本修改了Password算法，这里需要更新密码算法，便于使用客户端连接
ALTER user 'root'@'%' IDENTIFIED BY '123456' PASSWORD EXPIRE NEVER
ALTER user 'root'@'%' IDENTIFIED WITH mysql_native_password BY '123456'
# 刷新权限
FLUSH PRIVILEGES
```

新建hive数据库

```
create database db_hive;    --这个hive数据库与hive-site.xml中localhost:3306/hive的hive对应，用来保存hive元数据
```

5. 执行db_hive数据库表初始化命令：

```
schematool -dbType mysql -initSchema
```

6. 执行命令`hive`

```
hive
```

# 4 Zookeeper

1. 安装

```
tar -zxf apache-zookeeper-3.6.2-bin.tar.gz -C /usr/local/
cd /usr/local/
mv apache-zookeeper-3.6.2-bin zookeeper
```

2. 设置环境变量

```
vim /etc/profile
export ZOOKEEPER_HOME=/usr/local/zookeeper
export PATH=$ZOOKEEPER_HOME/bin:$PATH
# 立即生效
source /etc/profile
```

3. 修改配置文件,/conf进入文件夹对zoo_sample.cfg 文件重命名为zoo.cfg

```
cp /usr/local/zookeeper/conf/zoo_sample.cfg
/usr/local/zookeeper/conf/zoo.cfg
mkdir  /usr/local/zookeeper/dataDir
/usr/local/zookeeper/dataLogDir
vim /usr/local/zookeeper/conf/zoo.cfg
# 修改dataDir并添加dataLogDi配置
dataDir=/usr/local/zookeeper/dataDir
dataLogDir=/usr/local/zookeeper/dataLogDir
```

4. 启动zookeeper

```
#进入bin目录,键入命令启动服务
./zkServer.sh start
```

# 5 HBase

官网查看对应版本：https://hbase.apache.org/book.html#basic.prerequisites

下载地址：https://www.apache.org/dyn/closer.lua/hbase/2.2.6/hbase-2.2.6-bin.tar.gz

1. 安装

```
tar -zxvf hbase-2.2.6-bin.tar.gz -C /usr/local   # 解压到/usr/local中
cd /usr/local/
mv hbase-2.2.6 hbase        # 将文件夹名改为hbase
```

2. 配置环境变量

```
vim /etc/profile
export HBASE_HOME=/usr/local/hbase
export PATH=$HBASE_HOME/bin:$PATH
# 立即生效
source /etc/profile
```

3. 修改 /conf/hbase-env.sh文件

```
vi /usr/local/hbase/conf/hbase-env.sh
# 添加
export JAVA_HOME=/usr/local/java/jdk1.8.0_261
export HBASE_MANAGES_ZK=false  #其中HBASE_MANAGES_ZK=false表示我们使用自己安装zookeeper集群而不是hbase自带的zookeeper集群
```

4. 修改/conf/hbase-site.xml文件

```
<configuration>
  <property>
```

```
    <name>hbase.rootdir</name> <!-- hbase存放数据目录 -->
    <value>hdfs://hadoop-master:9000/hbase</value>
    <!-- 端口要和Hadoop的fs.defaultFS端口一致-->
  </property>
  <property>
    <name>hbase.cluster.distributed</name>
    <!-- 是否分布式部署 为 false 表示单机模式，为 true 表示分布式模式。-->
    <value>ture</value>
  </property>
  <property>
    <name>hbase.zookeeper.quorum</name>
    <!-- zookooper 服务启动的节点，只能为奇数个
    2181是zookeeper默认端口号，你可以自行根据你的端口号添加，默认的端口号加不加都无     所谓。-->
    <value>hadoop-master,hadoop-worker1,hadoop-worker2</value>
  </property>
  <property>
    <!--zookooper配置、日志等的存储位置，必须为以存在 -->
    <name>hbase.zookeeper.property.dataDir</name>
    <value>/usr/local/zookeeper/dataDir</value>
  </property>
  <property><!--hbase web 端口 -->
    <name>hbase.master.info.port</name>
    <value>16010</value>
  </property>
  <!-- 超时时间 -->
    <property>
        <name>zookeeper.session.timeout</name>
        <value>120000</value>
    </property>
    <!--防止服务器时间不同步出错 -->
```

```
    <property>
        <name>hbase.master.maxclockskew</name>
        <value>150000</value>
    </property>
    <property>
        <name>hbase.tmp.dir</name>
        <value>/usr/local/hadoop/hbase/tmp</value>
      </property>
</configuration>
```

5. 修改 regionservers 文件，删除localhost，添加region server 的ip

```
vim conf/regionservers
# 添加
hadoop-master
hadoop-worker1
hadoop-worker2
```

6. 将HBase安装包分发到其他节点
7. 启动hbase

```
start-hbase.sh
```