
目 录

| | |
|---------------------------------|----|
| Hadoop2 四种集群部署方案..... | 1 |
| 1. Hadoop 伪分布式方案..... | 1 |
| 1.1 实验环境准备..... | 1 |
| 1.2 安装 hadoop 并设置其所需的环境变量..... | 1 |
| 1.3 创建运行 Hadoop 进程的用户和相关目录..... | 2 |
| 1.4 配置 hadoop..... | 3 |
| 1.5 格式化 HDFS | 7 |
| 1.6 启动 Hadoop..... | 7 |
| 1.7 Web UI 概览 | 9 |
| 2. Hadoop 分布式集群部署方案..... | 15 |
| 2.1 实验环境准备..... | 15 |
| 2.2 安装 hadoop 并设置其所需的环境变量..... | 15 |
| 2.3 创建运行 Hadoop 进程的用户和相关目录..... | 19 |
| 2.4 配置 Hadoop..... | 20 |
| 2.5 配置各 slave 节点 | 24 |
| 2.6 格式化 HDFS | 26 |
| 2.7 启动 hadoop 集群..... | 26 |
| 2.8 验证..... | 27 |
| 3. Spark 集群部署 | 27 |
| 3.1 实验环境准备..... | 27 |
| 3.2 安装 Spark 并配置相应环境变量 | 28 |
| 3.3 创建运行 Spark 进程的用户和相关目录 | 29 |
| 3.4 配置 spark..... | 29 |
| 3.5 配置各 slave 节点 | 30 |
| 3.6 启动集群..... | 30 |
| 3.7 验证 spark 集群..... | 31 |
| 3.8 实例..... | 31 |

| | |
|------------------------------|----|
| 3.9 关闭 Spark 集群 | 35 |
| 4. Storm 集群部署 | 35 |
| 5. Hadoop 的 HA 分布式集群部署 | 35 |
| 6. Spark 的 HA 分布式集群部署 | 35 |
| 7. storm HA 集群部署 | 35 |

Don't Copy

Hadoop2 四种集群部署方案

主要有以下四种部署方案：

- 伪分布式
- 真正分布式
- HA 分布式
- HA 分布式+

1. Hadoop 伪分布式方案

1.1 实验环境准备

- CentOS7.1 虚拟机，x86_64 位系统，2G RAM，40GB 硬盘容量
- IP 地址：192.168.59.150/24

1.2 安装 hadoop 并设置其所需的环境变量

1. 首先关闭防火墙和 SeLinux

```
# 清除 iptables 规则

iptables -F

iptables -X

iptables -Z

chkconfig iptables off # 永久关闭

systemctl stop firewalld

systemctl disable firewalld #开机禁用

setenforce 0
```

2. 然后检查是否安装上 java1.7.0+,若未安装执行下面步骤

```
yum install java-1.8.0-openjdk java-1.8.0-openjdk-devel -y
```

然后检查 java 的版本号

```
[root@node ~]# java -version

openjdk version "1.8.0_161"

OpenJDK Runtime Environment (build 1.8.0_161-b14)

OpenJDK 64-Bit Server VM (build 25.161-b14, mixed mode)
```

并且将 java 写进环境变量中,这里环境变量有多种方式,可以写进/etc/profile

文件中，但是一般地，运维人员喜欢单独存放在`/etc/profile.d/`文件夹下，新建一个`java.sh`。

这里在`/etc/profile.d/`下新建一个文件，`java.sh`，并且内容如下：

```
export JAVA_HOME=/usr
```

下面是我从搭建的 `ftp` 服务器中获取的安装包（这步可以省略）

```
[root@node ~]# lftp
```

口令:

```
lftp shine@192.168.59.1:~> cd 7.x86_64/hadoop2/
```

```
lftp shine@192.168.59.1:/7.x86_64/hadoop2> get hadoop-2.6.2.tar.gz
```

```
195515434 bytes transferred in 5 seconds (34.87M/s)
```

```
lftp shine@192.168.59.1:/7.x86_64/hadoop2> bye
```

然后解压 `hadoop2.6.x` 安装包至指定目录下，

```
[root@node ~]# mkdir -pv /bdapps/
```

```
[root@node ~]# tar xf hadoop-2.6.2.tar.gz -C /bdapps/
```

```
[root@node ~]# ln -sv /bdapps/hadoop-2.6.2 /bdapps/hadoop
```

```
"/bdapps/hadoop" -> "/bdapps/hadoop-2.6.2"
```

接下来需要单独编辑环境配置文件`/etc/profile.d/hadoop.sh`，定义类似如下环境环境变量，设定 `Hadoop` 的运行环境。

```
export HADOOP_PREFIX="/bdapps/hadoop"
```

```
export PATH=$PATH:$HADOOP_PREFIX/bin:$HADOOP_PREFIX/sbin
```

```
export HADOOP_COMMON_HOME=${HADOOP_PREFIX}
```

```
export HADOOP_HDFS_HOME=${HADOOP_PREFIX}
```

```
export HADOOP_MAPRED_HOME=${HADOOP_PREFIX}
```

```
export HADOOP_YARN_HOME=${HADOOP_PREFIX}
```

1.3 创建运行 `Hadoop` 进程的用户和相关目录

1. 创建用户和组：

出于安全等目的，通常需要用特定的用户来运行 `hadoop` 不同的守护进程，例如，以 `hadoop` 为组，分别用三个用户 `yarn`、`hdfs` 和 `mapred` 来运行相应的进程。

```
[root@node ~]# groupadd hadoop
```

```
[root@node ~]# useradd -g hadoop hdfs
[root@node ~]# useradd -g hadoop yarn
[root@node ~]# useradd -g hadoop mapred
```

2. 创建数据和日志目录:

Hadoop 需要不同权限的数据和日志目录，这里以/data/hadoop/hdfs 为 hdfs 数据存储目录。

```
[root@node ~]# mkdir -pv /data/hadoop/hdfs/{nn,snn,dn}
[root@node ~]# chown -R hdfs:hadoop /data/hadoop/hdfs/
[root@node ~]# mkdir -p /var/log/hadoop/yarn
[root@node ~]# chown -R yarn:hadoop /var/log/hadoop/yarn/
```

而后在 hadoop 的安装目录中创建 log 目录。并且修改 hadoop 所有文件的属主和属性。

```
[root@node ~]# cd /bdapps/hadoop/
[root@master hadoop]# mkdir logs
[root@master hadoop]# chown g+w logs/
[root@master hadoop]# chmod g+w logs/
[root@master hadoop]# chown -R yarn:hadoop ./*
```

1.4 配置 hadoop

1. etc/hadoop/core-site.xml

core-site.xml 文件包含了 NameNode 主机地址以及其监听 RPC 端口等信息，对于伪分布式模型的安装来说，其主机地址为 localhost。NameNode 默认使用的 RPC 端口为 8020。其简要的配置内容如下所示。

```
<configuration>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://localhost:8020</value>
    <final>true</final>
  </property>
</configuration>
```

2. etc/hadoop/hdfs-site.xml

hdfs-site.xml 主要用于配置 HDFS 相关的属性，例如复制因子（即数据块的副本数）、NN 和 DN 用于存储数据的目录等。数据块的副本数对于伪分布式的 Hadoop 应该为 1，而 NN 和 DN 用于存储的数据的目录为前面的步骤中专门为其创建的路径。另外，前面的步骤中也为 SNN 创建了相关的目录，这里也一并配置其为启用状态。

```
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>
  <property>
    <name>dfs.namenode.name.dir</name>
    <value>file:///data/hadoop/hdfs/nn</value>
  </property>
  <property>
    <name>dfs.datanode.data.dir</name>
    <value>file:///data/hadoop/hdfs/dn</value>
  </property>
  <property>
    <name>fs.checkpoint.dir</name>
    <value>file:///data/hadoop/hdfs/snn</value>
  </property>
  <property>
    <name>fs.checkpoint.edits.dir</name>
    <value>file:///data/hadoop/hdfs/snn</value>
  </property>
</configuration>
```

注意，如果需要其它用户对 hdfs 有写入权限，还需要在 hdfs-site.xml 添加一项属性定义。

```
<property>
```

```
<name>dfs.permissions</name>

<value>>false</value>

</property>
```

3. etc/hadoop/mapred-site.xml

mapred-site.xml 文件用于配置集群的 MapReduce framework, 此处应该指定使用 yarn, 另外的可用值还有 local 和 classic。mapred-site.xml 默认不存在, 但有模块文件 mapred-site.xml.template. 只需要将其复制 mapred-site.xml 即可。

```
[root@node ~]# cp etc/hadoop/mapred-site.xml.template etc/hadoop/mapred-site.xml
```

其配置示例如下面的内容。

```
<configuration>

<property>

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

<value>yarn</value>

</property>

</configuration>
```

4. etc/hadoop/yarn-site.xml

yarn-site.xml 用于配置 YARN 进程及 YARN 的相关属性。首先需要指定 ResourceManager 守护进程的主机和监听的端口, 对于伪分布式模型来讲, 其主机为 localhost, 默认的端口为 8032; 其次需要指定 ResourceManager 使用的 scheduler, 以及 NodeManager 的辅助服务。

一个简要的配置示例如下所示。

```
<configuration>

  <property>

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

    <value>localhost:8032</value>

  </property>

  <property>

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

    <value>localhost:8030</value>

  </property>
```

```
<property>
    <name>yarn.resourcemanager.resource-tracker.address</name>
    <value>localhost:8031</value>
</property>
<property>
    <name>yarn.resourcemanager.admin.address</name>
    <value>localhost:8033</value>
</property>
<property>
    <name>yarn.resourcemanager.webapp.address</name>
    <value>localhost:8088</value>
</property>
<property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
</property>
<property>
    <name>yarn.nodemanager.auxservices.mapreduce_shuffle.class</name>
    <value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>
<property>
    <name>yarn.resourcemanager.scheduler.class</name>
<value>org.apache.hadoop.yarn.server.resourcemanager.scheduler.capacity.CapacityScheduler</v
alue>
</property>
</configuration>
```

5. etc/hadoop/hadoop-env.sh 和 etc/hadoop/yarn-env.sh

Hadoop 的各守护进程依赖于 JAVA_HOME 环境变量，如果有类似于前面

步骤中通过 `/etc/profile.d/java.sh` 全局配置定义的 `JAVA_HOME` 变量即可正常使用。不过，如果想为 Hadoop 定义依赖到的特定 JAVA 环境，也可以编辑这两个脚本文件，为其 `JAVA_HOME` 取消注释并配置合适的值即可。此外，Hadoop 大多数守护进程默认使用的堆大小为 1GB，但现实应用中，可能需要对其各类进程的堆内存大小做出调整，这只需要编辑此两者文件中的相关环境变量值即可，

比如：

`HADOOP_HEAPSIZE`

`HADOOP_JOB_HISTORY_HEAPSIZE`

`JAVA_HEAP_SIZE`

`YARN_HEAP_SIZE`

在这里就不做出修改了

6. slaves 文件

`slaves` 文件存储了当前集群的所有 `slave` 节点的列表，对于伪分布式模型，其文件内容仅应该为 `localhost`，这的确是这个文件的默认值。因此，伪分布式模型中，此文件的内容保持默认即可。

1.5 格式化 HDFS

在 HDFS 的 NN 启动之前需要先初始化其用于存储数据的目录。如果 `hdfs-site.xml` 中 `dfs.namenode.name.dir` 属性指定的目录不存在，格式化命令会自动创建之；如果事先存在，请确保其权限设置正确，此时格式操作会清除其内部的所有数据并重新建立一个新的文件系统。需要以 `hdfs` 用户的身份执行如下命令。

```
[root@master hadoop]# su - hdfs
```

```
[hdfs@master ~]$ hdfs namenode -format
```

其输出结果会有大量信息输出，如果显示出类似 “`INFO common.Storage: Storage directory /data/hadoop/hdfs/nn has been successfully formatted.`” 的结果表示格式化操作已然正确完成。

1.6 启动 Hadoop

Hadoop 2 的启动等操作可通过其位于 `sbin` 路径下的专用脚本进行：

```
NameNode: hadoop-daemon.sh (start|stop) namenode
```

DataNode: `hadoop-daemon.sh (start|stop) datanode`

Secondary NameNode: `hadoop-daemon.sh (start|stop) secondarynamenode`

ResourceManager: `yarn-daemon.sh (start|stop) resourcemanager`

NodeManager: `yarn-daemon.sh (start|stop) nodemanager`

HDFS 有三个守护进程：namenode、datanode 和 secondarynamenode，它们都可通过 `hadoop-daemon.sh` 脚本启动或停止。以 `hdfs` 用户执行相关的命令即可，如下所示。

```
[hdfs@master ~]$ hadoop-daemon.sh start namenode
```

```
[hdfs@master ~]$ hadoop-daemon.sh start secondarynamenode
```

```
[hdfs@master ~]$ hadoop-daemon.sh start datanode
```

上述三个命令均在执行完成后给出了一个日志信息保存指向的信息，但是，实际用于保存日志的文件是以 “.log” 为后缀的文件，而非以 “.out” 结尾。可通过日志文件中的信息来判断进程启动是否正常完成。如果所有进程正常启动，可通过 `jdk` 提供的 `jps` 命令来查看相关的 `java` 进程状态。

```
[hdfs@master ~]$ jps
```

```
3876 Jps
```

```
3766 SecondaryNameNode
```

```
3815 DataNode
```

```
3722 NameNode
```

另外，`hadoop-daemon.sh` 脚本的 `stop` 命令各分别用于停止此三个守护进程。

2. 启动 YARN 服务

YARN 有两个守护进程：`resourcemanager` 和 `nodemanager`，它们都可通过 `yarn-daemon.sh` 脚本启动或停止。以 `yarn` 用户执行相关的命令即可，如下所示。

```
[root@node ~]# su - hadoop
```

```
[yarn@master ~]$ yarn-daemon.sh start resourcemanager
```

```
starting resourcemanager, logging to /bdapps/hadoop/logs/yarn-yarn-resourcemanager-master.out
```

```
[yarn@master ~]$ yarn-daemon.sh start nodemanager
```

```
starting nodemanager, logging to /bdapps/hadoop/logs/yarn-yarn-nodemanager-master.out
```

上述两个命令均在执行完成后给出了一个日志信息保存指向的信息，但是，实际用于保存日志的文件是以“.log”为后缀的文件，而非以“.out”结尾。可通过日志文件中的信息来判断进程启动是否正常完成。如果所有进程正常启动，可通过 `jdk` 提供的 `jps` 命令来查看相关的 `java` 进程状态。

```
[yarn@master ~]$ jps
3957 ResourceManager
4200 NodeManager
4252 Jps
```

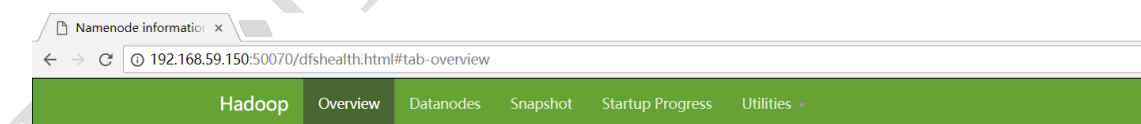
1.7 Web UI 概览

HDFS 和 YARN ResourceManager 各自提供了一个 Web 接口，通过这些接口可检查 HDFS 集

群以及 YARN 集群的相关状态信息。它们的访问接口分别为如下所求，具体使用中，需要将 `NameNodeHost` 和 `RresourceManagerHost` 分别改为其相应的主机地址。

```
HDFS-NameNode http://<NameNodeHost>:50070/
YARN-ResourceManager http://<ResourceManagerHost>:8088/
```

注意：`yarn-site.xml` 文件中 `yarn.resourcemanager.webapp.address` 属性的值如果定义为“localhost:8088”，则其 WebUI 仅监听于 127.0.0.1 地址上的 8088 端口。



Overview 'localhost:8020' (active)

| | |
|----------------|---|
| Started: | Thu Mar 15 10:51:59 CST 2018 |
| Version: | 2.6.2, r0cfd050febe4a30b1ee1551dcc527589509fb681 |
| Compiled: | 2015-10-22T00:42Z by jenkins from (detached from 0cfd050) |
| Cluster ID: | CID-8cacd4bc-6dda-4e70-bcb6-4fa44cd67244 |
| Block Pool ID: | BP-410834951-127.0.0.1-1521082102538 |

Summary



All Applications

Cluster

[About](#)
[Nodes](#)
[Applications](#)
[NEW](#)
[NEW_SAVING](#)
[SUBMITTED](#)
[ACCEPTED](#)
[RUNNING](#)
[FINISHED](#)
[FAILED](#)
[KILLED](#)
[Scheduler](#)

Tools

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 |
|----------------|--------------|--------------|----------------|--------------------|-------------|--------------|-----------------|-------------|--------------|-----------------|--------------|
| 0 | 0 | 0 | 0 | 0 | 0 B | 0 B | 0 B | 0 | 0 | 0 | 0 |

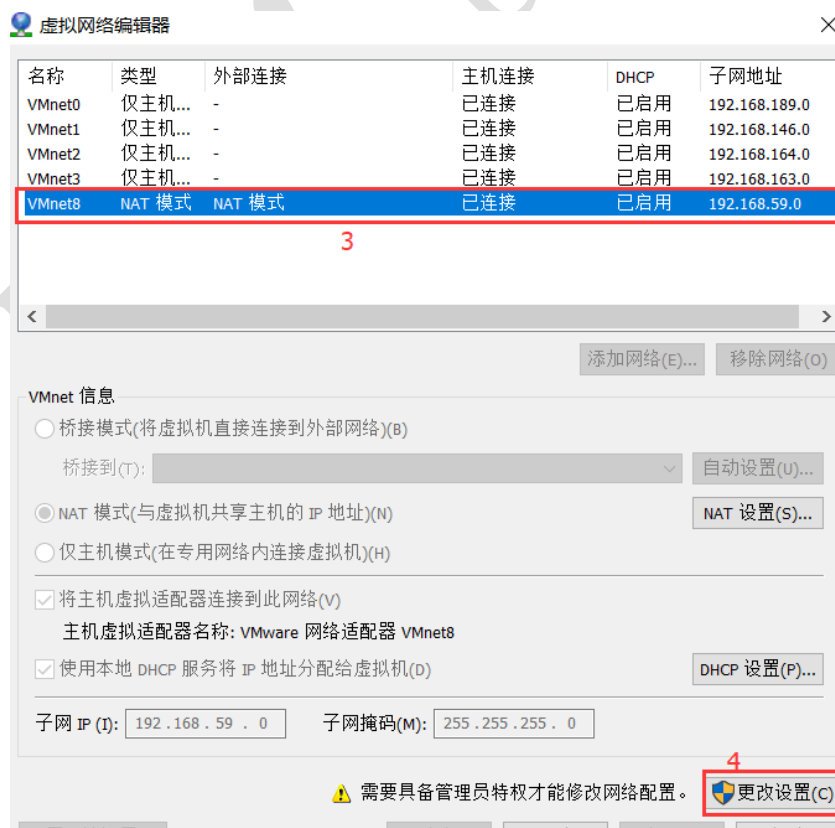
Show 20 entries

| ID | User | Name | Application Type | Queue | StartTime | FinishTime | State | FinalStatus | Progress |
|----|------|------|------------------|-------|-----------|------------|-------|-------------|----------|
|----|------|------|------------------|-------|-----------|------------|-------|-------------|----------|

No data available in table

Showing 0 to 0 of 0 entries

③ 如何通过 vmware NAT 转化功能，将 8088 端口的流量转发给客户端？
配置如下：





第七步 运行测试程序

Hadoop-YARN 自带了许多样例程序，它们位于 `hadoop` 安装路径下的 `share/hadoop/mapreduce/` 目录里，其中的 `hadoop-mapreduce-examples` 可用作 `mapreduce` 程序测试。

```
[root@node ~]# su - hdfs
[hdfs@node ~]$ yarn jar
/bdapps/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.6.2.jar
```

例如，运行 `pi` 程序，用 Monte Carlo 方法估算 $\text{Pi} (\pi)$ 值。`pi` 命令有两个参数，第一个参数是指要运行 `map` 的次数，第二个参数是指每个 `map` 任务取样的个数；而两数相乘即为总的取样数。其示例和输出结果如下所求。

```
[hdfs@node ~]$ yarn jar
/bdapps/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.6.2.jar pi 2 10
```

Number of Maps = 2

Samples per Map = 4

Wrote input for Map #0

Wrote input for Map #1

Starting Job

18/03/15 11:44:05 INFO client.RMProxy: Connecting to ResourceManager at localhost/127.0.0.1:8032

18/03/15 11:44:07 INFO input.FileInputFormat: Total input paths to process : 2

18/03/15 11:44:08 INFO mapreduce.JobSubmitter: number of splits:2

18/03/15 11:44:09 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1521084413130_0003

18/03/15 11:44:10 INFO impl.YarnClientImpl: Submitted application application_1521084413130_0003

18/03/15 11:44:10 INFO mapreduce.Job: The url to track the job: http://192.168.59.150:8088/proxy/application_1521084413130_0003/

18/03/15 11:44:10 INFO mapreduce.Job: Running job: job_1521084413130_0003

18/03/15 11:45:17 INFO mapreduce.Job: Job job_1521084413130_0003 running in uber mode : false

18/03/15 11:45:17 INFO mapreduce.Job: map 0% reduce 0%

18/03/15 11:45:28 INFO mapreduce.Job: map 50% reduce 0%

18/03/15 11:45:29 INFO mapreduce.Job: map 100% reduce 0%

18/03/15 11:45:36 INFO mapreduce.Job: map 100% reduce 100%

18/03/15 11:45:37 INFO mapreduce.Job: Job job_1521084413130_0003
completed successfully

18/03/15 11:45:38 INFO mapreduce.Job: Counters: 49

File System Counters

FILE: Number of bytes read=50

FILE: Number of bytes written=319584

FILE: Number of read operations=0

FILE: Number of large read operations=0

FILE: Number of write operations=0

HDFS: Number of bytes read=526

HDFS: Number of bytes written=215

HDFS: Number of read operations=11

HDFS: Number of large read operations=0

HDFS: Number of write operations=3

Job Counters

Launched map tasks=2

Launched reduce tasks=1

Data-local map tasks=2

Total time spent by all maps in occupied slots (ms)=19670

Total time spent by all reduces in occupied slots (ms)=4658

Total time spent by all map tasks (ms)=19670

Total time spent by all reduce tasks (ms)=4658

Total vcore-seconds taken by all map tasks=19670

Total vcore-seconds taken by all reduce tasks=4658

Total megabyte-seconds taken by all map tasks=20142080

Total megabyte-seconds taken by all reduce tasks=4769792

Map-Reduce Framework

Map input records=2

Map output records=4

Map output bytes=36
Map output materialized bytes=56
Input split bytes=290
Combine input records=0
Combine output records=0
Reduce input groups=2
Reduce shuffle bytes=56
Reduce input records=4
Reduce output records=0
Spilled Records=8
Shuffled Maps =2
Failed Shuffles=0
Merged Map outputs=2
GC time elapsed (ms)=431
CPU time spent (ms)=1910
Physical memory (bytes) snapshot=494571520
Virtual memory (bytes) snapshot=6297976832
Total committed heap usage (bytes)=301146112

Shuffle Errors

BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0

File Input Format Counters

Bytes Read=236

File Output Format Counters

Bytes Written=97

Job Finished in 93.251 seconds

Estimated value of Pi is 3.50000000000000000000

| 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 |
|----------------|--------------|--------------|----------------|--------------------|-------------|--------------|-----------------|-------------|--------------|-----------------|--------------|----------------------|
| 3 | 0 | 1 | 2 | 3 | 4 GB | 8 GB | 0 B | 3 | 8 | 0 | 1 | 0 |

| ID | User | Name | Application Type | Queue | StartTime | FinishTime | State | FinalStatus | Progress |
|--------------------------------|------|-----------------|------------------|---------|-------------------------------|-------------------------------|----------|-------------|-------------|
| application_1521084413130_0001 | hdfs | QuasiMonteCarlo | MAPREDUCE | default | Thu, 15 Mar 2018 03:37:13 GMT | Thu, 15 Mar 2018 03:44:26 GMT | FINISHED | SUCCEEDED | <div></div> |

2. Hadoop 分布式集群部署方案

2.1 实验环境准备

■ 4×CentOS7.1 虚拟机

- ◆ 1×master x86_64 位系统, 2G RAM, 40GB 硬盘容量
- ◆ 3×slave x86_64 位系统, 1.5G RAM, 40GB 硬盘容量

■ IP 地址分配

- ◆ 192.168.59.150/24 master
- ◆ 192.168.59.154/24-192.168.59.156/24 slave

分别为 dn1 dn2 dn3

2.2 安装 hadoop 并设置其所需的环境变量

以下都是针对所有的机器设定的, 包括 2.3 节.

1. 首先关闭防火墙和 SeLinux

```
# 清除 iptables 规则

iptables -F

iptables -X

iptables -Z

chkconfig iptables off # 永久关闭

systemctl stop firewalld

systemctl disable firewalld #开机禁用

setenforce 0
```

2. 然后检查是否安装上 java1.7.0+,若未安装, 在所有机器上执行下面步骤

```
yum install java-1.8.0-openjdk java-1.8.0-openjdk-devel -y
```

然后检查 java 的版本号

```
[root@node ~]# java -version
```

```
openjdk version "1.8.0_161"
```

```
OpenJDK Runtime Environment (build 1.8.0_161-b14)
```

```
OpenJDK 64-Bit Server VM (build 25.161-b14, mixed mode)
```

并且将 java 写进环境变量中,这里环境变量有多种方式,可以写进/etc/profile 文件中,但是一般地,运维人员喜欢单独存放在/etc/profile.d/文件夹下,新建一个 java.sh.

这里在/etc/profile.d/下新建一个文件, java.sh, 并且内容如下:

```
export JAVA_HOME=/usr
```

```
[root@master~]# for i in {dn1,dn2,dn3};do scp /etc/profile.d/java.sh
root@${i}:/etc/profile.d/ ; done
```

下面是我从搭建的 ftp 服务器中获取的安装包(这步可以省略)

```
[root@node ~]# lftp
```

口令:

```
lftp shine@192.168.59.1:~> cd 7.x86_64/hadoop2/
```

```
lftp shine@192.168.59.1:/7.x86_64/hadoop2> get hadoop-2.6.2.tar.gz
```

```
195515434 bytes transferred in 5 seconds (34.87M/s)
```

```
lftp shine@192.168.59.1:/7.x86_64/hadoop2> bye
```

然后解压 hadoop2.6.x 安装包至指定目录下,

```
[root@node ~]# mkdir -pv /bdapps/
```

```
[root@node ~]# tar xf hadoop-2.6.2.tar.gz -C /bdapps/
```

```
[root@node ~]# ln -sv /bdapps/hadoop-2.6.2 /bdapps/hadoop
```

```
"/bdapps/hadoop" -> "/bdapps/hadoop-2.6.2"
```

接下来需要单独编辑环境配置文件/etc/profile.d/hadoop.sh, 定义类似如下环境环境变量, 设定 Hadoop 的运行环境。

```
export HADOOP_PREFIX="/bdapps/hadoop"
```

```
export PATH=$PATH:$HADOOP_PREFIX/bin:$HADOOP_PREFIX/sbin
```

```
export HADOOP_COMMON_HOME=${HADOOP_PREFIX}
```

```
export HADOOP_HDFS_HOME=${HADOOP_PREFIX}
```

```
export HADOOP_MAPRED_HOME=${HADOOP_PREFIX}
```

```
export HADOOP_YARN_HOME=${HADOOP_PREFIX}
```

```
[root@master ~]# for i in {dn1,dn2,dn3};do scp /etc/profile.d/hadoop.sh
root@${i}:/etc/profile.d/ ; done
```

4. [提前做]为了方便起见，建议利用密钥来连接各个 slave，方便后面的文件传输。

```
[root@master ~]# ssh-keygen -t rsa -b 4096 -P "
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
b6:3d:02:8f:89:7b:63:26:cf:bf:38:e2:0b:39:2c:59 root@master
The key's randomart image is:
+--[ RSA 4096]-----+
|           |
|           |
|           |
|           |
|  E    . S   |
|+ . . * o    |
|o = . o + o   |
| . 000*. . .  |
|   .+O++o.    |
+-----+
```

```
[root@master ~]# for i in {dn1,dn2,dn3};do ssh-copy-id -i .ssh/id_rsa.pub root@${i}; done
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are
already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to
```

install the new keys

root@dn1's password:

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'root@dn1'"

and check to make sure that only the key(s) you wanted were added.

/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed

/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys

root@dn2's password:

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'root@dn2'"

and check to make sure that only the key(s) you wanted were added.

/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed

/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys

root@dn3's password:

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'root@dn3'"

and check to make sure that only the key(s) you wanted were added.

```
[root@master ~]# for i in {dn1,dn2,dn3};do scp /etc/hosts root@${i}:/etc/ ; done
```

| | | | | |
|-------|------|-----|---------|-------|
| hosts | 100% | 266 | 0.3KB/s | 00:00 |
| hosts | 100% | 266 | 0.3KB/s | 00:00 |
| hosts | 100% | 266 | 0.3KB/s | 00:00 |

```
[root@master ~]# cat /etc/hosts

127.0.0.1    localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6

192.168.69.150    master
192.168.59.154    dn1
192.168.59.155    dn2
192.168.59.156    dn3
```

2.3 创建运行 Hadoop 进程的用户和相关目录

每台主机都要做。

1. 创建用户和组：

出于安全等目的，通常需要用特定的用户来运行 `hadoop` 不同的守护进程，例如，以 `hadoop` 为组，分别用三个用户 `yarn`、`hdfs` 和 `mapred` 来运行相应的进程。

```
[root@node ~]# groupadd hadoop

[root@node ~]# useradd -g hadoop hdfs

[root@node ~]# useradd -g hadoop yarn

[root@node ~]# useradd -g hadoop mapred

[root@node ~]# groupadd hadoop

[root@node ~]# for user in yarn hdfs mapred;do useradd -g hadoop ${user} &&
echo 'hadoop' | passwd --stdin ${user};done
```

2. 创建数据和日志目录：

Hadoop 需要不同权限的数据和日志目录，这里以 `/data/hadoop/hdfs` 为 `hdfs` 数据存储目录。

```
[root@node ~]# mkdir -pv /data/hadoop/hdfs/{nn,snn,dn}

[root@node ~]# chown -R hdfs:hadoop /data/hadoop/hdfs/

[root@node ~]# mkdir -p /var/log/hadoop/yarn

[root@node ~]# chown -R yarn:hadoop /var/log/hadoop/yarn/
```

而后在 **hadoop** 的安装目录中创建 **log** 目录。并且修改 **hadoop** 所有文件的属主和属性。

```
[root@node ~]# cd /bdapps/hadoop/

[root@master hadoop]# mkdir logs

[root@master hadoop]# chown g+w logs/

[root@master hadoop]# chmod g+w logs/

[root@master hadoop]# chown -R yarn:hadoop ./*
```

2.4 配置 Hadoop

1. etc/hadoop/core-site.xml

core-site.xml 文件包含了 **NameNode** 主机地址以及其监听 **RPC** 端口等信息，对于伪分布式模型的安装来说，其主机地址为 **master**。**NameNode** 默认使用的 **RPC** 端口为 **8020**。其简要的配置内容如下所示。

```
<configuration>

  <property>

    <name>fs.defaultFS</name>

    <value>hdfs://master:8020</value>

    <final>true</final>

  </property>

</configuration>
```

2. etc/hadoop/hdfs-site.xml

hdfs-site.xml 主要用于配置 **HDFS** 相关的属性，例如复制因子（即数据块的副本数）、**NN** 和 **DN** 用于存储数据的目录等。数据块的副本数为所需要冗余的数值 **2**，而 **NN** 和 **DN** 用于存储的数据的目录为前面的步骤中专门为其创建的路径。另外，前面的步骤中也为 **SNN** 创建了相关的目录，这里也一并配置其为启用状态。

```
<configuration>

  <property>

    <name>dfs.replication</name>

    <value>2</value>
```

```

</property>

<property>

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

<value>file:///data/hadoop/hdfs/nn</value>

</property>

<property>

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

<value>file:///data/hadoop/hdfs/dn</value>

</property>

<property>

<name>fs.checkpoint.dir</name>

<value>file:///data/hadoop/hdfs/snn</value>

</property>

<property>

<name>fs.checkpoint.edits.dir</name>

<value>file:///data/hadoop/hdfs/snn</value>

</property>

</configuration>

```

注意，如果需要其它用户对 hdfs 有写入权限，还需要在 hdfs-site.xml 添加一项属性定义。

```

<property>

<name>dfs.permissions</name>

<value>>false</value>

</property>

```

3. etc/hadoop/mapred-site.xml

mapred-site.xml 文件用于配置集群的 MapReduce framework，此处应该指定使用 yarn，另外的可用值还有 local 和 classic。mapred-site.xml 默认不存在，但有模块文件 mapred-site.xml.template。只需要将其复制 mapred-site.xml 即可。

```
[root@node ~]# cp etc/hadoop/mapred-site.xml.template etc/hadoop/mapred-site.xml
```

其配置示例如下面的内容。

```
<configuration>

<property>

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

<value>yarn</value>

</property>

</configuration>
```

4. etc/hadoop/yarn-site.xml

yarn-site.xml 用于配置 YARN 进程及 YARN 的相关属性。首先需要指定 ResourceManager 守护进程的主机和监听的端口，对于分布式模型来讲，其主机为 master，默认的端口为 8032；其次需要指定 ResourceManager 使用的 scheduler，以及 NodeManager 的辅助服务。

一个简要的配置示例如下所示。

```
<configuration>

  <property>

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

    <value>master:8032</value>

  </property>

  <property>

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

    <value>master:8030</value>

  </property>

  <property>

    <name>yarn.resourcemanager.resource-tracker.address</name>

    <value>master:8031</value>

  </property>

  <property>

    <name>yarn.resourcemanager.admin.address</name>

    <value>master:8033</value>

  </property>
```



```

<property>
    <name>yarn.resourcemanager.webapp.address</name>
    <value>master:8088</value>
</property>
<property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
</property>
<property>
    <name>yarn.nodemanager.auxservices.mapreduce_shuffle.class</name>
    <value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>
<property>
    <name>yarn.resourcemanager.scheduler.class</name>
<value>org.apache.hadoop.yarn.server.resourcemanager.scheduler.capacity.CapacityScheduler</value>
</property>
</configuration>

```

5. etc/hadoop/hadoop-env.sh 和 etc/hadoop/yarn-env.sh

Hadoop 的各守护进程依赖于 JAVA_HOME 环境变量，如果有类似于前面步骤中通过 /etc/profile.d/java.sh 全局配置定义的 JAVA_HOME 变量即可正常使用。不过，如果想为 Hadoop 定义依赖到的特定 JAVA 环境，也可以编辑这两个脚本文件，为其 JAVA_HOME 取消注释并配置合适的值即可。此外，Hadoop 大多数守护进程默认使用的堆大小为 1GB，但现实应用中，可能需要对其各类进程的堆内存大小做出调整，这只需要编辑此两者文件中的相关环境变量值即可，

比如：

HADOOP_HEAPSIZE

HADOOP_JOB_HISTORY_HEAPSIZE

JAVA_HEAP_SIZE

YARN_HEAP_SIZE

在这里就不做出修改了

6. slaves 文件

slaves 文件存储了当前集群的所有 slave 节点的列表，对于分布式模型，其文件内容应该为所有 slave 的主机名，这的确是这个文件的默认值。因此，伪分布式模型中，此文件的内容保持默认即可。

```
dn1
dn2
dn3
```

2.5 配置各 slave 节点

slave 节点的配置与 master 节点相同，只是启动的服务不同，因此，在各 slave 节点创建所需的用户、组和目录、设置好权限并安装 hadoop 之后，将 master 节点的各配置文件直接复制到各 slave 节点 hadoop 配置文件目录，再为其设置好各环境变量即可。所有 slave 节点的安装配置过程均相同。

(1) 以下步骤在各 slave 节点操作

设置好 JAVA_HOME 变量，而后解压安装包至指定目录下

```
mkdir -pv /bdapps/
tar xf hadoop-2.6.2.tar.gz -C /bdapps/
ln -sv /bdapps/hadoop-2.6.2 /bdapps/hadoop
```

编辑环境配置文件/etc/profile.d/hadoop.sh，定义类似如下环境环境变量，设定 Hadoop 的运行环境。

```
export HADOOP_PREFIX="/bdapps/hadoop"
export PATH=$PATH:$HADOOP_PREFIX/bin:$HADOOP_PREFIX/sbin
export HADOOP_COMMON_HOME=${HADOOP_PREFIX}
export HADOOP_HDFS_HOME=${HADOOP_PREFIX}
export HADOOP_MAPRED_HOME=${HADOOP_PREFIX}
export HADOOP_YARN_HOME=${HADOOP_PREFIX}
```

创建用户和组：

出于安全等目的，通常需要用特定的用户来运行 `hadoop` 不同的守护进程，例如，以 `hadoop` 为组，分别用三个用户 `yarn`、`hdfs` 和 `mapred` 来运行相应的进程。

```
groupadd hadoop

for user in hdfs yarn mapred; do useradd -g hadoop $user && echo 'YOUR_PASSWORD' |
passwd --stdin $user; done
```

创建数据和日志目录：

`Hadoop` 需要不同权限的数据和日志目录，这里以 `/data/hadoop/hdfs` 为 `hdfs` 数据存储目录。一般来说，从节点只用得到 `dn` 子目录。

```
mkdir -pv /data/hadoop/hdfs/{nn,snn,dn}

chown -R hdfs:hadoop /data/hadoop/hdfs/

mkdir -p /var/log/hadoop/yarn

chown yarn:hadoop /var/log/hadoop/yarn
```

而后，在 `hadoop` 的安装目录中创建 `logs` 目录，并修改 `hadoop` 所有文件的属主和属组。

```
cd /bdapps/hadoop/

mkdir logs

chmod g+w logs

chown -R yarn:hadoop ./*
```

(2) 下面的步骤在 `master` 节点操作

```
[yarn@master ~]$ ssh-keygen -t rsa -P ""

[yarn@master ~]$ for i in 1 2 3; do ssh-copy-id -i .ssh/id_rsa.pub yarn@dn${i};done

[hdfs@master ~]$ ssh-keygen -t rsa -P ""

[hdfs@master ~]$ for i in 1 2 3; do ssh-copy-id -i .ssh/id_rsa.pub hdfs@dn${i};done

[mapred@master ~]$ ssh-keygen -t rsa -P ""

[mapred@master ~]$ for i in 1 2 3; do ssh-copy-id -i .ssh/id_rsa.pub mapred@dn${i};done
```

复制配置文件至各 `slave` 节点。

```
for slave in node2 node3 node4; do scp
/bdapps/hadoop/etc/hadoop/{core-site.xml,hdfs-site.xml,yarn-site.xml,mapred-site.xml}
```

```
{slave}:/bdapps/hadoop/etc/hadoop/; done
```

2.6 格式化 HDFS

与伪分布式模式相同,在 HDFS 集群的 NN 启动之前需要先初始化其用于存储数据的目录。如果 hdfs-site.xml 中 dfs.namenode.name.dir 属性指定的目录不存在,格式化命令会自动创建之;如果事先存在,请确保其权限设置正确,此时格式操作会清除其内部的所有数据并重新建立一个新的文件系统。需要以 hdfs 用户的身份在 master 节点执行如下命令。

```
[hdfs@master~]$ hdfs namenode -format
```

2.7 启动 hadoop 集群

启动 Hadoop-YARN 集群的方法有两种:一是在各节点分别启动需要启动的服务,二是在 master 节点启动整个集群。

(1) 分别启动

master 节点需要启动 HDFS 的 NameNode 服务,以及 YARN 的 ResourceManager 服务。根据我们前述的配置,启动 hdfs 的相关服务需要以 hdfs 用户的身份进行,而启动 yarn 相关的服务需要 yarn 用户的身份进行。

```
[root@master~]# su - hdfs -c 'hadoop-daemon.sh start namenode'
```

```
[root@master~]# su - yarn -c 'yarn-daemon.sh start resourcemanager'
```

各 slave 节点需要启动 HDFS 的 DataNode 服务,以及 YARN 的 NodeManager 服务。

```
[root@master~]# su - hdfs -c 'hadoop-daemon.sh start datanode'
```

```
[root@master~]# su - yarn -c 'yarn-daemon.sh start nodemanager'
```

(2) 在 master 节点控制整个集群

集群规模较大时,分别启动各节点的服务过于繁琐和低效,为此,hadoop 专门提供了 start-dfs.sh 和 stop-dfs.sh 来启动及停止整个 hdfs 集群,以及 start-yarn.sh 和 stop-yarn.sh 来启动及停止整个 yarn 集群。

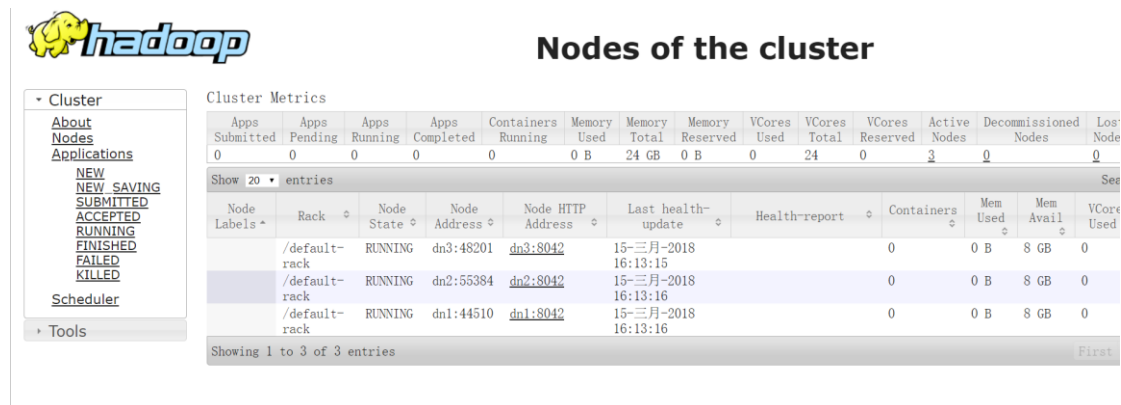
```
[root@master~]# su - hdfs -c 'start-dfs.sh'
```

```
[root@master~]# su - yarn -c 'start-yarn.sh'
```

较早版本的 hadoop 会提供 start-all.sh 和 stop-all.sh 脚本来统一控制 hdfs 和 mapreduce,但 hadoop 2.0 及之后的版本不建议再使用此种方式。

2.8 验证

集群启动完成后，可在各节点以 `jps` 命令等验证各进程是否正常运行，也可以通过 Web UI 来检查集群的运行状态。



1 files and directories, 0 blocks = 1 total filesystem object(s).

Heap Memory used 33.34 MB of 49.02 MB Heap Memory. Max Heap Memory is 966.69 MB.

Non Heap Memory used 38.03 MB of 38.94 MB Committed Non Heap Memory. Max Non Heap Memory is -1 B.

| | |
|--|-------------------------------|
| Configured Capacity: | 112.34 GB |
| DFS Used: | 12 KB |
| Non DFS Used: | 5.33 GB |
| DFS Remaining: | 107 GB |
| DFS Used%: | 0% |
| DFS Remaining%: | 95.25% |
| Block Pool Used: | 12 KB |
| Block Pool Used%: | 0% |
| DataNodes usages% (Min/Median/Max/stdDev): | 0.00% / 0.00% / 0.00% / 0.00% |
| Live Nodes | 3 (Decommissioned: 0) |
| Dead Nodes | 0 (Decommissioned: 0) |
| Decommissioning Nodes | 0 |
| Number of Under-Replicated Blocks | 0 |

3. Spark 集群部署（Spark On Yarn）

首先在这里，Spark 集群在 hadoop 集群的基础上进行部署和安装的

3.1 实验环境准备

- 4×CentOS7.1 虚拟机
 - ◆ 1×master x86_64 位系统，2G RAM，40GB 硬盘容量
 - ◆ 3×slave x86_64 位系统，1.5G RAM，40GB 硬盘容量
- IP 地址分配

- ◆ 192.168.59.150/24 master
- ◆ 192.168.59.154/24-192.168.59.156/24 slave

分别为 dn1 dn2 dn3

3.2 安装 Spark 并配置相应的环境变量

保证安装了 java 和 scala，其中 java 的安装以及相关的环境变量配置已经在前面说过了。下面说一下 scala 的安装和配置环境变量。

```
[root@master ~]# lftp
命令:
lftp shine@192.168.59.1:~> cd spark/
lftp shine@192.168.59.1:/spark> ls
-rwxrwxrwx  1 owner  group      19741785 Mar 14 10:28 scala-2.12.4.tgz
drwxrwxrwx  1 owner  group           0 Mar 14 17:41 spark
-rwxrwxrwx  1 owner  group    224121109 Mar 14 10:30 spark-2.3.0-bin-hadoop2.6.tgz
lftp shine@192.168.59.1:/spark> get scala-2.12.4.tgz spark-2.3.0-bin-hadoop2.6.tgz
243862894 bytes transferred in 24 seconds (9.89M/s)
Total 2 files transferred
lftp shine@192.168.59.1:/spark> bye
```

下载完成之后，解压到指定的文件夹下，这里以/bdapps/为例，然后编辑环境变量。

```
[root@master ~]# tar xf scala-2.12.4.tgz -C /bdapps/

[root@master ~]# ln -sv /bdapps/scala-2.12.4 /bdapps/scala

"/bdapps/scala" -> "/bdapps/scala-2.12.4"
```

编辑/etc/profile.d/scala.sh

```
export SCALA_HOME="/bdapps/scala"

export PATH=$PATH:$SCALA_HOME/bin
```

在这里我省略了一步，其实需要到 spark 的官方网站上去下载相应的安装包。

<https://spark.apache.org/downloads.html>

1. Choose a Spark release:
2. Choose a package type:
3. Download Spark: [spark-2.3.0-bin-hadoop2.7.tgz](#)
4. Verify this release using the [2.3.0 signatures and checksums](#) and [project release KEYS](#).

然后解压 spark2.3 安装包至指定目录下，

```
[root@master ~]# tar xf spark-2.3.0-bin-hadoop2.6.tgz -C /bdapps/

[root@master ~]# ln -sv /bdapps/spark-2.3.0-bin-hadoop2.6 /bdapps/spark

"/bdapps/spark" -> "/bdapps/spark-2.3.0-bin-hadoop2.6"
```

编辑 spark 的运行环境/etc/profile.d/spark.sh

```
export SPARK_HOME="/bdapps/spark/"
export $PATH=$PATH:$SPARK_HOME/bin:$SPARK_HOME/sbin
export SCALA_LIBRARY_PATH=${SPARK_HOME}/lib
export SPARK_LAUNCH_WITH_SCALA=0
export STANDALONE_SPARK_MASTER_HOST=192.168.59.150
export SPARK_MASTER_IP=192.168.59.150
export HADOOP_CONF_DIR=/bdapps/hadoop/etc/hadoop
```

3.3 创建运行 Spark 进程的用户和相关目录

在这里使用 yarn 用户来管理 spark 即可。

```
[root@master ~]# cd /bdapps/spark
```

```
[root@master spark]# ls
```

```
bin    conf    data    examples  jars    kubernetes  LICENSE  licenses
```

```
NOTICE  python  R    README.md  RELEASE  sbin  yarn
```

```
[root@master spark]# mkdir work
```

```
[root@master spark]# chmod g+w work/
```

```
[root@master spark]# mkdir logs
```

```
[root@master spark]# chmod g+w logs/
```

```
[root@master spark]# chown -R yarn:hadoop ./*
```

3.4 配置 spark

1. spark-env.sh

这是 spark 的环境变量配置，可以指定特定的运行环境变量，由于有的前面已经配置过了，这里只配置有用的。也可以不配置。

JAVA_HOME: Java 安装目录

SCALA_HOME: Scala 安装目录

HADOOP_HOME: hadoop 安装目录

HADOOP_CONF_DIR: hadoop 集群的配置文件的目录

SPARK_MASTER_IP: spark 集群的 Master 节点的 ip 地址

SPARK_WORKER_MEMORY: 每个 worker 节点能够最大分配给 executors

的内存大小

SPARK_WORKER_CORES: 每个 worker 节点所占有的 CPU 核数目

SPARK_WORKER_INSTANCES: 每台机器上开启的 worker 节点的数目

```
export SPARK_WORKER_MEMORY=1g
export SPARK_WORKER_CORES=1
export SPARK_WORKER_INSTANCES=1
```

2. slaves

这个文件时用户自己创建的，和 hadoop 一样。

```
dn1
dn2
dn3
```

3.5 配置各 slave 节点

和配置 master 一样

```
[root@dn1 ~]# tar xf scala-2.12.4.tgz -C /bdapps/
[root@dn1 ~]# ln -sv /bdapps/scala-2.12.4 /bdapps/scala
"/bdapps/scala" -> "/bdapps/scala-2.12.4"
[root@dn1 ~]# tar xf spark-2.3.0-bin-hadoop2.6.tgz -C /bdapps/
[root@dn1 ~]# ln -sv /bdapps/spark-2.3.0-bin-hadoop2.6 /bdapps/spark
"/bdapps/spark" -> "/bdapps/spark-2.3.0-bin-hadoop2.6"
[root@master ~]# for i in {dn1,dn2,dn3};do scp /etc/profile.d/{scala.sh,spark.sh}
root@${i}:/etc/profile.d;/done
[root@master ~]# for i in {dn1,dn2,dn3};do scp /bdapps/spark/conf/spark-env.sh
root@${i}:/bdapps/spark/conf;/done
[root@master ~]# for i in {dn1,dn2,dn3};do ssh root@${i} 'mkdir -pv
/bdapps/spark/{work,logs} && chmod g+w /bdapps/spark/{work,logs} && chown -R
yarn:hadoop /bdapps/spark/*';done
```

3.6 启动集群

先启动 hadoop 集群，然后启动 spark 集群

```
[root@master sbin]# su - hdfs
[hdfs@master ~]$ start-dfs.sh
```



```
[root@master ~]# su - yarn
```

```
[yarn@master ~]$ start-yarn.sh
```

```
[yarn@master ~]$ cd /bdapps/spark/sbin/
```

```
[yarn@master sbin]$ ./start-all.sh
```

```
starting      org.apache.spark.deploy.master.Master,      logging      to
/bdapps/spark/logs/spark-yarn-org.apache.spark.deploy.master.Master-1-master.out
```

```
dn3:    starting      org.apache.spark.deploy.worker.Worker,      logging      to
/bdapps/spark/logs/spark-yarn-org.apache.spark.deploy.worker.Worker-1-dn3.out
```

```
dn1:    starting      org.apache.spark.deploy.worker.Worker,      logging      to
/bdapps/spark/logs/spark-yarn-org.apache.spark.deploy.worker.Worker-1-dn1.out
```

```
dn2:    starting      org.apache.spark.deploy.worker.Worker,      logging      to
/bdapps/spark/logs/spark-yarn-org.apache.spark.deploy.worker.Worker-1-dn2.out
```

3.7 验证 spark 集群

打开 webui 地址: <http://192.168.59.150:8080/>

Spark Master at spark://master:7077

URL: spark://master:7077
 REST URL: spark://master:6066 (cluster mode)
 Alive Workers: 3
 Cores in use: 3 Total, 0 Used
 Memory in use: 3.0 GB Total, 0.0 B Used
 Applications: 0 Running, 0 Completed
 Drivers: 0 Running, 0 Completed
 Status: ALIVE

Workers (3)

| Worker Id | Address | State | Cores | Memory |
|--|----------------------|-------|------------|-----------|
| worker-20180315201231-192.168.59.156-42637 | 192.168.59.156:42637 | ALIVE | 1 (0 Used) | 1024.0 MB |
| worker-20180315201232-192.168.59.154-43308 | 192.168.59.154:43308 | ALIVE | 1 (0 Used) | 1024.0 MB |
| worker-20180315201232-192.168.59.155-52501 | 192.168.59.155:52501 | ALIVE | 1 (0 Used) | 1024.0 MB |

Running Applications (0)

| Application ID | Name | Cores | Memory per Executor | Submitted Time | User | St |
|----------------|------|-------|---------------------|----------------|------|----|
|----------------|------|-------|---------------------|----------------|------|----|

Completed Applications (0)

| Application ID | Name | Cores | Memory per Executor | Submitted Time | User | St |
|----------------|------|-------|---------------------|----------------|------|----|
|----------------|------|-------|---------------------|----------------|------|----|

3.8 实例

1. 集群管理器

Spark 可以运行在各种集群管理器上, 并通过集群管理器访问集群中的机器。

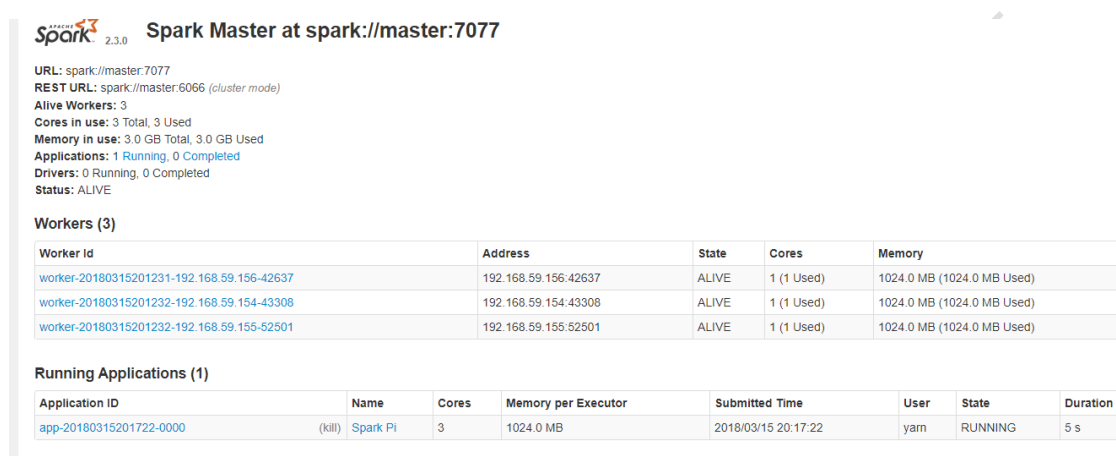
这里通过简单的示例介绍两种独立集群管理器和 Hadoop Yarn 集群管理器。

2. 独立集群管理器

向独立集群管理器提交应用，需要把 `spark://master:7077` 作为主节点参数递给 `spark-submit`。

在 Shell 中输入如下命令：

```
[yarn@master spark]$ bin/spark-submit --class org.apache.spark.examples.SparkPi --master spark://master:7077 examples/jars/spark-examples_2.11-2.3.0.jar 100 2>&1 | grep "Pi is roughly"
```



Spark Master at spark://master:7077

URL: spark://master:7077
 REST URL: spark://master:6066 (cluster mode)
 Active Workers: 3
 Cores in use: 3 Total, 3 Used
 Memory in use: 3.0 GB Total, 3.0 GB Used
 Applications: 1 Running, 0 Completed
 Drivers: 0 Running, 0 Completed
 Status: ALIVE

Workers (3)

| Worker Id | Address | State | Cores | Memory |
|--|----------------------|-------|------------|----------------------------|
| worker-20180315201231-192.168.59.156-42637 | 192.168.59.156:42637 | ALIVE | 1 (1 Used) | 1024.0 MB (1024.0 MB Used) |
| worker-20180315201232-192.168.59.154-43308 | 192.168.59.154:43308 | ALIVE | 1 (1 Used) | 1024.0 MB (1024.0 MB Used) |
| worker-20180315201232-192.168.59.155-52501 | 192.168.59.155:52501 | ALIVE | 1 (1 Used) | 1024.0 MB (1024.0 MB Used) |

Running Applications (1)

| Application ID | Name | Cores | Memory per Executor | Submitted Time | User | State | Duration |
|-------------------------|-----------------|-------|---------------------|---------------------|------|---------|----------|
| app-20180315201722-0000 | (kill) Spark Pi | 3 | 1024.0 MB | 2018/03/15 20:17:22 | yarn | RUNNING | 5 s |

结果：

```
[yarn@master spark]$ bin/spark-submit --class org.apache.spark.examples.SparkPi --master spark://master:7077 examples/jars/spark-examples_2.11-2.3.0.jar 100 2>&1 | grep "Pi is roughly"
```

Pi is roughly 3.1419507141950715

也可以用 `spark-shell` 连接到独立集群管理器上。

hdfs 用户：在 Shell 中输入如下命令：

```
[hdfs@master ~]$ hdfs dfs -put /bdapps/spark/README.md /cd /usr/local/spark/
```

yarn 用户：

```
bin/spark-shell --master spark://master:7077
```

Spark Master at spark://master:7077

URL: spark://master:7077
 REST URL: spark://master:6066 (cluster mode)
 Alive Workers: 3
 Cores in use: 3 Total, 3 Used
 Memory in use: 3.0 GB Total, 3.0 GB Used
 Applications: 1 Running, 1 Completed
 Drivers: 0 Running, 0 Completed
 Status: ALIVE

Workers (3)

| Worker id | Address | State | Cores | Memory |
|--|----------------------|-------|------------|----------------------------|
| worker-20180315201231-192.168.59.156-42637 | 192.168.59.156:42637 | ALIVE | 1 (1 Used) | 1024.0 MB (1024.0 MB Used) |
| worker-20180315201232-192.168.59.154-43308 | 192.168.59.154:43308 | ALIVE | 1 (1 Used) | 1024.0 MB (1024.0 MB Used) |
| worker-20180315201232-192.168.59.155-52501 | 192.168.59.155:52501 | ALIVE | 1 (1 Used) | 1024.0 MB (1024.0 MB Used) |

Running Applications (1)

| Application ID | Name | Cores | Memory per Executor | Submitted Time | User | State | Duration |
|-------------------------|--------------------|-------|---------------------|---------------------|------|---------|----------|
| app-20180315202519-0001 | (kill) Spark shell | 3 | 1024.0 MB | 2018/03/15 20:25:19 | yarn | RUNNING | 2 s |

Completed Applications (1)

在 scala 控制台上输入：

```
scala> val textFile = sc.textFile("hdfs://192.168.59.150:8020/README.md")

textFile: org.apache.spark.rdd.RDD[String] = hdfs://192.168.59.150:8020/README.md
MapPartitionsRDD[1] at textFile at <console>:24
```

```
scala> textFile.count()

res0: Long = 103

scala> textFile.first()

res1: String = # Apache Spark
```

Hadoop Yarn 管理器

向 Hadoop Yarn 集群管理器提交应用，需要把 `yarn-cluster` 作为主节点参数递给 `spark-submit`。

在 Shell 中输入如下命令：

这时需要修改 `hdfs-site.xml`

```
<property>

  <name>dfs.permissions</name>

  <value>>false</value>

</property>
```

否则会报错误！

```

2018-03-15 20:33:02 WARN MetricsSystem:66 - Stopping a MetricsSystem that is not running
2018-03-15 20:33:02 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint:54 - OutputCommitCoordinator stopped:
2018-03-15 20:33:02 INFO SparkContext:54 - Successfully stopped SparkContext
Exception in thread "main" org.apache.hadoop.security.AccessControlException: Permission denied: user=yarn, access=WRITE, inode="/":hdfs:supergroup:drwxr-xr-x
    at org.apache.hadoop.hdfs.server.namenode.FSPermissionChecker.checkPermission(FSPermissionChecker.java:274)
    at org.apache.hadoop.hdfs.server.namenode.FSPermissionChecker.checkPermission(FSPermissionChecker.java:257)
    at org.apache.hadoop.hdfs.server.namenode.FSPermissionChecker.checkPermission(FSPermissionChecker.java:238)
    at org.apache.hadoop.hdfs.server.namenode.FSPermissionChecker.checkPermission(FSPermissionChecker.java:179)
    at org.apache.hadoop.hdfs.server.namenode.FSNamesystem.checkPermission(FSNamesystem.java:6545)
    at org.apache.hadoop.hdfs.server.namenode.FSNamesystem.checkPermission(FSNamesystem.java:6527)
    at org.apache.hadoop.hdfs.server.namenode.FSNamesystem.checkAncestorAccess(FSNamesystem.java:6479)
    at org.apache.hadoop.hdfs.server.namenode.FSNamesystem.mkdirsInternal(FSNamesystem.java:4290)
    at org.apache.hadoop.hdfs.server.namenode.FSNamesystem.mkdirsInt(FSNamesystem.java:4260)
    at org.apache.hadoop.hdfs.server.namenode.FSNamesystem.mkdirs(FSNamesystem.java:4239)

```

```

bin/spark-submit --class org.apache.spark.examples.SparkPi --master yarn-cluster
examples/jars/spark-examples_2.11-2.0.2.jar

```

```

2018-03-15 20:56:13 INFO Client:54 - Application report for application_1521118373384_0001 (state: FINISHED)
2018-03-15 20:56:13 INFO Client:54 -
    client token: N/A
    diagnostics: N/A
    ApplicationMaster host: 192.168.59.156
    ApplicationMaster RPC port: 0
    queue: default
    start time: 1521118494253
    final status: SUCCEEDED
    tracking URL: http://master:8088/proxy/application_1521118373384_0001/
    user: yarn
2018-03-15 20:56:13 INFO ShutdownHookManager:54 - Shutdown hook called

```

复制结果地址到浏览器，点击查看 Logs，再点击 stdout，即可查看结果，如下图所示：

Master 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 | Lost Nodes | Unhealthy Nodes | Rebooted Nodes |
|----------------|--------------|--------------|----------------|--------------------|-------------|--------------|-----------------|-------------|--------------|-----------------|--------------|----------------------|------------|-----------------|----------------|
| 0 | 0 | 0 | 1 | 0 | 0 B | 24 GB | 0 B | 0 | 24 | 0 | 3 | 0 | 0 | 0 | 0 |

Showing 1 to 1 of 1 entries

Search:

| ID | User | Name | Application Type | Queue | StartTime | FinishTime | State | FinalStatus | Progress | Tracking UI | Blacklisted Nodes |
|--------------------------------|------|-----------------------------------|------------------|---------|-------------------------------|-------------------------------|----------|-------------|----------|-------------|-------------------|
| application_1521118373384_0001 | yarn | org.apache.spark.examples.SparkPi | SPARK | default | Thu, 15 Mar 2018 12:54:54 GMT | Thu, 15 Mar 2018 12:56:13 GMT | FINISHED | SUCCEEDED | | History | N/A |

Showing 1 to 1 of 1 entries

First Previous 1 Next Last

→ 192.168.59.150:8088/cluster/app/application_1521118373384_0001

NEW SAVING
SUBMITTED
ACCEPTED
RUNNING
FINISHED
FAILED
KILLED

Scheduler

Tools

Application Type: SPARK

Application Tags:

YarnApplicationState: FINISHED

FinalStatus Reported by AM: SUCCEEDED

Started: 15-三月-2018 20:54:54

Elapsed: 1mins, 18sec

Tracking URL: History

Diagnostics:

Total Resource Preempted: <memory:0

Total Number of Non-AM Containers Preempted: 0

Total Number of AM Containers Preempted: 0

Resource Preempted from Current Attempt: <memory:0

Number of Non-AM Containers Preempted from Current Attempt: 0

Aggregate Resource Allocation: 302253 MB

Show 20 entries

| Attempt ID | Started | Node | Logs |
|-----------------------------------|-------------------------------|-----------------|------|
| attempt_1521118373384_0001_000001 | Thu, 15 Mar 2018 12:54:54 GMT | http://dn3:8042 | Logs |

Showing 1 to 1 of 1 entries

← → ↻ ⓘ 192.168.59.156:8042/node/containerlogs/container_1521118373384_0001_01_000001/yarn



Logs for
container_1521118373384_0001_01_000001

▼ ResourceManager
RM Home

▶ NodeManager

▶ Tools

[stderr : Total file length is 538 bytes.](#)
[stdout : Total file length is 16874 bytes.](#)

3.9 关闭 Spark 集群

```
[yarn@master spark]$ sbin/stop-all.sh
```

4. Storm 集群部署

5. Hadoop 的 HA 分布式集群部署

6. Spark 的 HA 分布式集群部署

7. storm HA 集群部署