大数据系列（3）—Hadoop集群完全分布式环境搭建 - 指尖流淌 - 博客园

笔记本： 技术
创建时间： 2018/7/20 16:38
作者： Jason
标签： hadoop
URL： https://www.cnblogs.com/zhijianliutang/p/5756738.html

大数据系列（3）——Hadoop集群完全分布式环境搭建

前言

上一篇我们讲解了Hadoop单节点的安装，并且已经通过VMware安装了一台CentOS 6.8的Linux系统，咱们本篇的目标就是要配置一个真正的完全分布式的Hadoop集群，闲言少叙，进入本篇的正题。

技术准备

VMware虚拟机、CentOS 6.8 64 bit

安装流程

我们先来回顾上一篇我们完成的单节点的Hadoop环境配置，已经配置了一个CentOS 6.8 并且完成了java运行环境的搭建，Hosts文件的配置、计算机名等诸多细节。

其实完成这一步之后我们就已经完成了Hadoop集群的搭建的一半的工作了，因为我们知道通过虚拟机搭建所搭建的好处就是直接拷贝机器。多台同步进行操作，减少分别配置的时间消耗浪费。这也是虚拟化技术所带来的优势。

下面，咱们进去分布式系统的详细操作过程。

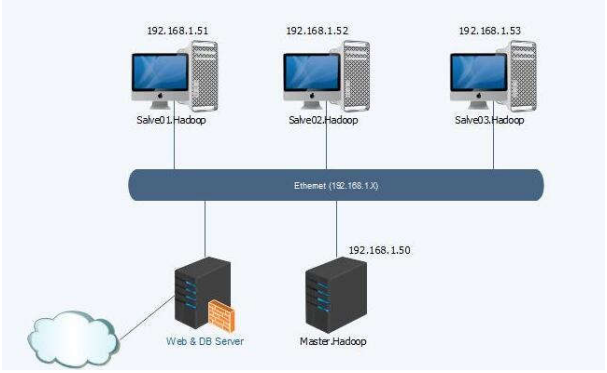
1、首先需要在VMWare中将之前创建的单实例的计算机进行拷贝。

这里根据之前第一篇文章的规划，我们至少需要再克隆出三台计算机，作为DataNode数据节点的数据存储。之前的上一台机器作为Master主节点进行管理。

机器名称	IP地址	角色	OS
Master.Hadoop	192.168.1.50	Master	CentOS6.8
Salve01.Hadoop	192.168.1.51	Salve1	CentOS6.8
Salve02.Hadoop	192.168.1.52	Salve2	CentOS6.8
Salve03.Hadoop	192.168.1.53	Salve3	CentOS6.8
MySQLServer01	102.168.1.100	MySQLServer	Ubuntu

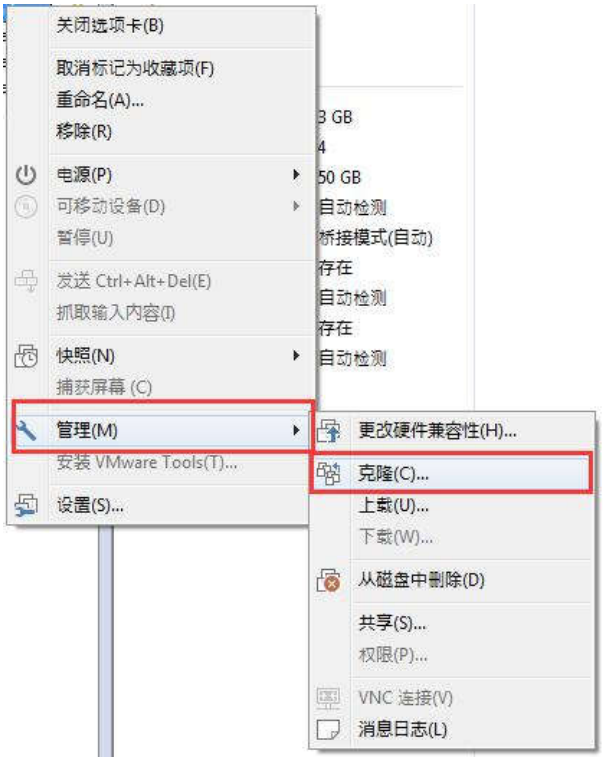
这里先来梳理一下整个Hadoop集群的物理架构图，大家有一个直接的观念和认识，上表中已经和明确了，总共需要5台服务器来使用，四台用来搭建Hadoop集群使用，另外一台（可选）作为MySQL等外围管理Hadoop集群来使用。

我们在开发的时候一般也是直接通过连接外围的这台机器来管理Hadoop整个集群。

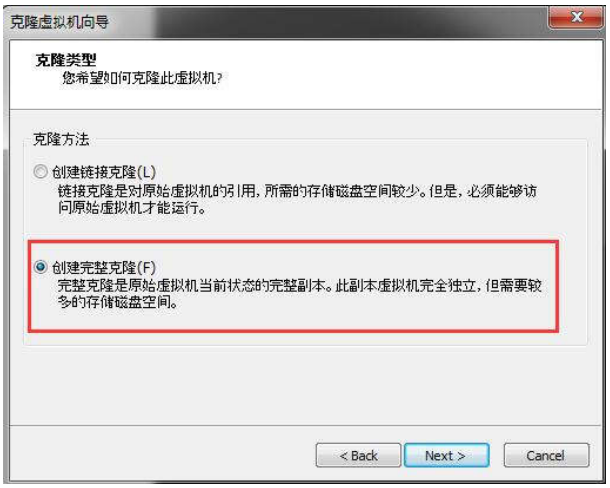


根据上面的物理规划图应该对整个架构有一个清晰的认识了，好，咱们进行实操。

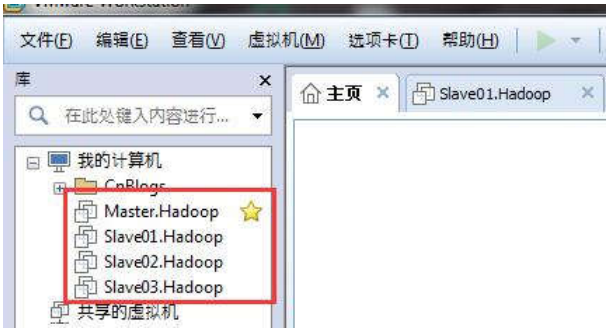
关于在VMWare中进行虚拟机的拷贝是一个比较简单的过程。截图如下：



然后，就是下一步就行了，这里需要记住的是，一定要选择克隆一个完整的而不是快照。



然后，根据计算机名输入机器名就可以了。克隆之后的机器如下：



2、配置各个Slave节点的机器信息。

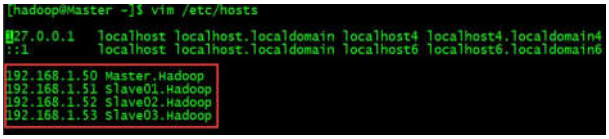
关于各个Slave服务器的配置基本分为如下基本部分：

- 首先需要手动更改各个从节点的计算机名和Hosts文件（必须！）
- 然后配置各个从节点的内存值，在第一篇的文章中我已经分析过了，这里可以将这里的内存值设置的比Master节点少点，（土豪公司忽略！）
- 最后配置的就是存储了，这个自己根据之前的计算公式计算出来就可以了。

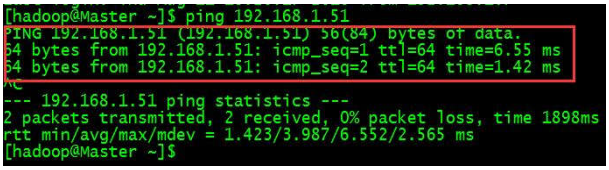
首先，进入各个机器中更改Hosts文件和计算机名，在上一篇文章我已经介绍过了，大家可以上一篇翻阅，这里直接写出脚本如下：

```
vim /etc/sysconfig/network
vim /etc/hosts
```

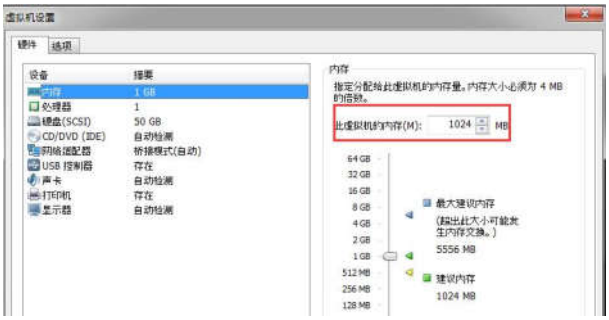
计算机名和Hosts配置文件按照之前规划完成就行了，同样网络的IP地址也是按照规划设置成固定的地址。



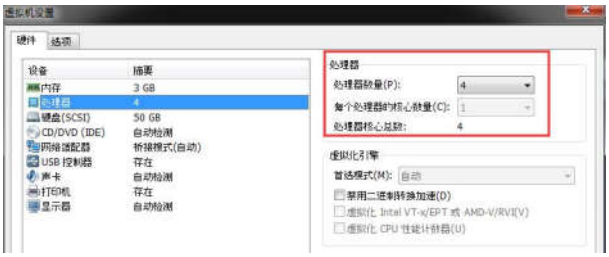
在配置完这一切之后，重启完各个机器之后，确保各个节点之间可以ping 通（重点！！！）。



然后剩下的内存配置，直接关闭掉虚拟机，在VMWare中进行设置就可以了，很简单。



这里根据需要自行调整，然后如果可以的话，尽量将主节点Master的CUP处理器设置成多路多核，这样设置的原因，我第一篇文章中就已经详细分析过了。



至此，各个服务器的基础配置已经完成了。

经过上面一系列流程，大家是不是发现通过虚拟机拷贝这种方式省去了好多额外的配置时间，比如：装操作系统、下载Hadoop安装包、搭建Java环境等。

3、配置SSH无密码配置。

先来解释下SSH的概念和用途；

SSH 为 Secure Shell 的缩写，由 IETF 的网络小组（Network Working Group）所制定；SSH 为建立在应用层和传输层基础上的安全协议。SSH 是目前较可靠，专为远程登录会话和其他网络服务提供安全性的协议。利用 SSH 协议可以有效防止远程管理过程中的信息泄露问题。SSH最初是UNIX系统上的一个程序，后来又迅速扩展到其他操作平台。SSH在正确使用时可弥补网络中的漏洞。SSH客户端适用于多种平台。几乎所有UNIX平台—包括HP-UX、Linux、AIX、Solaris、Digital UNIX、Irix，以及其他平台，都可运行SSH。

上面就是SSH的官方含义了，摘自百度百科。

下面，我来总结下SSH在Hadoop集群中的用途。

所谓的SSH简单一句话就是：同一用户无密码登录到各台机器。 其实，就是所有的Hadoop集群中作为分布式的一个计算框架，需要对各个节点的服务进行操作，而操作的过程中需要统一由一个相同的用户进行操作，但是同一用户登录不同的服务器都需要密码或者密钥进行身份验证。为了避免这个验证过程就使用了统一的一种安全协议：SSH。

其实，SSH的原理很简单，就是提前将统一用户的密码进行加密形成密钥进行分发，然后分发到各个服务器中，各个服务器对这个秘钥加入到当前的系统用户组中，这样这个用户登录的时候就不需要输入密码进行登录操作了。

希望，我上面的讲解各位看官能看明白里面的含义。

下面咱们来实际操作：

- 首先进行sshd的配置文件的修改，去掉默认注释，开启SSH验证功能（以root用户进行操作）。

```
vim /etc/ssh/sshd_config
```

```
RSAAuthentication yes
PubkeyAuthentication yes
AuthorizedKeysFile      .ssh/authorized_keys
#AuthorizedKeysCommand none
#AuthorizedKeysCommandRunAs nobody
```

将上面的这三行数据的注释“#”去掉进行，保存。这里记住了！所有的机器都要这么依次进行设置。

简要的解释下上面三行数据的含义：1、第一个RSAAuthentication是指开启SSH验证，2、PubkeyAuthetication是指可以通过公钥进行验证，3、AuthorizedkeysFile则指的的是公钥存放的位置。

记住，完成配置之后，重启该服务，脚本如下：

```
/sbin/service sshd restart
```

```
[root@Master hadoop]#  /sbin/service sshd restart
Stopping sshd:                                             [  OK  ]
Starting sshd:                                             [  OK  ]
```

可以验证下，比如这里我这里直接SSH登录本机系统：

```
ssh localhost
```

```
[root@Master hadoop]# ssh localhost
The authenticity of host 'localhost (::1)' can't be established.
RSA key fingerprint is 6d:cf:62:6f:ae:39:a7:57:c3:b3:39:2a:c9:61:5b:b9.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'localhost' (RSA) to the list of known hosts.
root@localhost's password:
```

这里可以看到，丫让我输入密码，所以说这里只是开启了SSH验证，但是没有生成密钥，进行设置。

- 加工生成证书公私钥，分发到各个服务器（以Hadoop用户操作）。

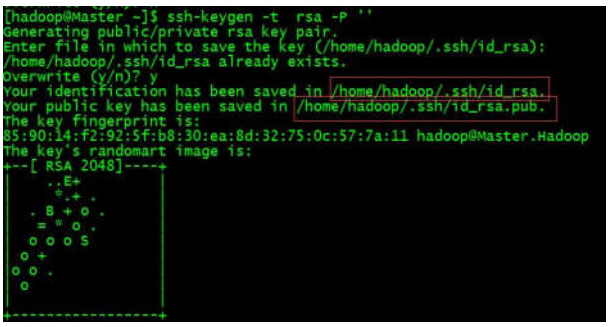
这个步骤就是上面我分析的过程，我们需要在Master节点上生成Hadoop用户的公钥，然后将这个公钥分发

给各个slave节点，然后这样在Master机器上就可以用Hadoop无密码登录到各个salve机器上面了。

步骤如下：

```
ssh-keygen -t  rsa -P ''
```

这里的-P后面'P'是大写的。



上面我用红框勾出的路径就是公钥和私钥生成的默认路径。

然后，下一步就是将这个公钥复制到各个slave节点中去、

通过以下Linux命令进行远程文件的复制，脚本命令如下：

```
scp ~/.ssh/id_rsa.pub 远程用户名@远程服务器IP:~/
```

我们这里面要复制的公钥文件存在默认的路径"**/home/hadoop/.ssh**"，所以执行的命令就是

```
scp ~/.ssh/id_rsa.pub hadoop@192.168.1.51:~/
```

然后，我们需要登录192.168.1.51的salve01的机器将刚才生成的公钥加入的本地的权限验证组里面去。

```
 cat ~/id_rsa.pub >> ~/.ssh/authorized_keys
```

上面的命令是要在Slave01的机器上进行的，并且使用hadoop用户今次那个操作。

最后，我们来回到Master机器上面进行，ssh验证。

SSH验证的命令很简单，格式如下：

```
SSH  <远程IP && 域名>
```

所以，这里咱们在master机器上登录slave01机器上实验下，看是否还需要进行密码输入。

```
ssh slave01.hadoop
```

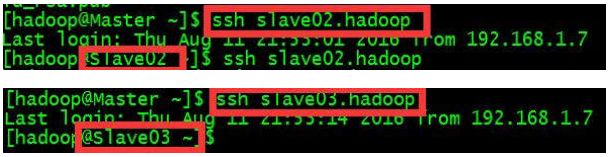
通过上面的命令窗口可以看到，我们已经成功的从Master机器上面无密码的登录到Slave01机器上面了。那么说明刚才的配置生效了。

- 参照上面的步骤将各个Slave节点配置完成。

这里需要配置的剩下的两个Slave节点进行无密码登录了，详细的操作过程参照上面的流程就可以了，需要的注意的是：在Master生成密钥只需要生成一次就可以了，不要再次生成！因为每次生成以为着所有的节点

都需要重新配置。

配置完成之后的效果，就是要保证在master机器上面的hadoop用户能够无需密码的登录到各个slave节点上进行操作就可以。



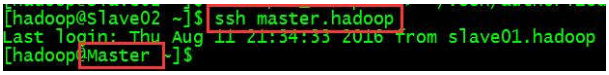
经过上面的操作，已经确保我们的Master机器可以毫无障碍的操作各个子节点Slave了。

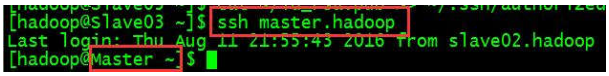
- 参照上面的步骤将各个Slave节点SSH到Master机器。

  我们知道，经过上面的一系列操作之后，我们的Master主节点可以顺利的操控各个Slave节点了，但是，这里需要注意的是，为了保证各个Slave机器和Master机器进行通信。

  需要保证各个Slave节点能够无密码登录Master机器，操作步骤如上面。

  这么操作的理由很简单，各个Slave子节点干完Master分配的任务之后，需要有权限反馈至他们的老大Master!





好了，到此，我们已经完成了整个集群的SSH配置了。

这里再次强调，上面的步骤要一定完成验证，要不以后的Hadoop操作会很出现各种诡异的问题，让你措手不及，这都是经验！！

4、配置Hadoop集群配置。

好了，到此我们需要对各个机器上面的Hadoop进行配置了。我们知道这里的所有的机器都是从一台机器上面的拷贝过来，因为我们在这个机器上面已经安装了单实例的Hadoop，参照上一篇文章。

那么，下一步的步骤就是将这个单节点的配置成一个真正的分布式集群，充分利用我们刚才搭建的几台Server进行性能的最大发挥。

这里的配置不是很多，只需要更改一下几个文件就可以了。

- 首先进行slaves文件的配置，指定该集群的各个Slave节点的位置（以hadoop用户进行操作）。

  这个只需要在Master的机器上面进行就可以了，当然，如果不介意可以保持所有的机器上面的Hadoop配置一样就可以了。执行命令如下

  ```
  vim /usr/hadoop/hadoop-2.6.4/etc/hadoop/slaves
  ```

  然后，将各个Slave的IP或者机器名写入就可以了，一台机器一行数据。这里我写的是IP.

  这样就可以了。

- 然后，在更改hdfs-site.xml文件中的dfs.replication属性值。

关于这个值我之前已经解释过了，因为我们现在不是单台机器了，所以将这个节点更改成3或者更大的数，因为咱们就四台机器，所以这里就配置成3可以了。记住：只能是奇数！

```
vim /usr/hadoop/hadoop-2.6.4/etc/hadoop/hdfs-site.xml
```

```
<property>
 <name>dfs.replication</name>
 <value>3</value>
</property>
```

这里需要注意的是，所有的机器都要这样配置。

5、启动Hadoop集群，验证是否成功。

到此，我们基本完成了一个Hadoop完全分布式的集群配置。下面的内容就是我们来验证一下是否可用。

验证方式就很简单了，首先我们先来执行一个HDFS格式的命令，这个在上一篇我们已经分析过了，因为，咱们这里改成完全分布式的集群，所以这里需要重新格式。

```
bin/hadoop namenode -format
```

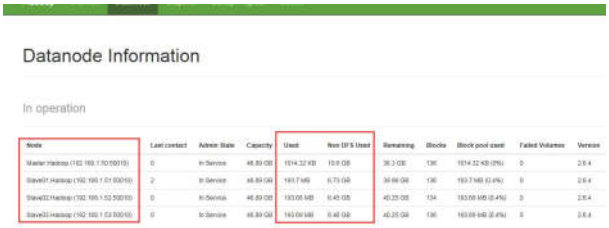
- 首先，我们来验证一下整个集群的HDFS是否正常可用。

  启动整个集群的HDFS，在Master机器上面，用hadoop用户操作，命令如下：

  ```
  start-dfs.sh
  ```

我们通过浏览器来查看整个集群的HDFS状态，地址为：http://192.168.1.50:50070/dfshealth.html#tab-overview

可以看到，当前我们的Hadoop集群的HDFS集群已经成功启动，然后我们来看整个集群的存储和节点数；



从上面的截图我们可以看到，当前的集群存在四个DataNode节点，就是刚才我们配置的SIave文件的IP.这说明我们配置的集群HDFS能够正常运行。

- 然后，我们来验证一下整个集群的YARN分布式计算框架是否正常可用。

  同样的方式，我们首先来启动Yarn.脚本如下：

```
start-yarn.sh
```

我们通过浏览器来查看整个集群的Hadoop集群状态，地址为：http://192.168.1.50:8088/

可以看到，当前的Hadoop集群已经存在四个正在运行的节点，而且跑的很Happy.后面的文章我将跟大家分析如何使用这个Hadoop集群。

结语

此篇先到此吧，关于Hadoop大数据集群的搭建后续依次介绍，比如利用Zookeeper搭建Hadoop高可用平台、Map-Reducer层序的开发、Hive产品的数据分析、Spark的应用程序的开发、Hue的集群坏境的集成和运维、Sqoop2的数据抽取等，有兴趣的童鞋可以提前关注。

本篇主要介绍了搭建一个完全分布式的Hadoop集群，后面我们会逐渐完善它，我会教你如何一步步的使用完全分布式的Hadoop集群，然后教你如何使用它，骚年...不要捉急...让思维飞一会...

有问题可以留言或者私信，随时恭候有兴趣的童鞋加大数据平台深入研究。共同学习，一起进步。

文章的最后给出上一篇的基础篇：

大数据系列（1）——Hadoop集群坏境搭建配置

大数据系列（2）——Hadoop集群坏境CentOS安装

如果您看了本篇博客,觉得对您有所收获，请不要吝啬您的“**推荐**”。

分类: Hadoop安装

指尖流淌
关注 - 47
粉丝 - 925
+加关注

11 0

« 上一篇：大数据系列（2）——Hadoop集群坏境CentOS安装
» 下一篇：大数据系列（4）——Hadoop集群VSFTP和SecureCRT安装配置

posted @ 2016-08-11 22:45 指尖流淌 阅读(7232) 评论(4) 编辑 收藏