

# 尚硅谷大数据技术之 Hadoop（入门）

（作者：尚硅谷大数据研发部）

版本：V3.0

## 第 1 章 大数据概论

### 1.1 大数据概念



#### 大数据概念



大数据（Big Data）：指**无法在一定时间范围内**用常规软件工具进行捕捉、管理和处理的数据集合，是需要新处理模式才能具有更强的决策力、洞察发现力和流程优化能力的**海量、高增长率和多样化的信息资产**。

主要解决，海量数据的**存储**和海量数据的**分析计算**问题。

按顺序给出数据存储单位：bit、Byte、KB、MB、GB、**TB**、**PB**、**EB**、ZB、YB、BB、NB、DB。

1Byte = 8bit 1K = 1024Byte 1MB = 1024K

1G = 1024M **1T = 1024G** **1P = 1024T**



### 1.2 大数据特点（4V）



#### 大数据特点



#### 1、Volume（大量）

截至目前，人类生产的所有**印刷材料的数据量是200PB**，而历史上全人类总共说过的话的数据量大约是**5EB**。当前，典型个人计算机硬盘的容量为TB量级，而一些**大企业的数据量已经接近EB量级**。



### 2、Velocity (高速)

这是大数据区别于传统数据挖掘的最显著特征。根据IDC的“数字宇宙”的报告,预计到2025年,全球数据使用量将达到163ZB。在如此海量的数据面前,处理数据的效率就是企业的生命。

天猫双十一: 2017年3分01秒, 天猫交易额超过100亿  
2019年1分36秒, 天猫交易额超过100亿



### 3、Variety (多样)

这种类型的多样性也让数据被分为结构化数据和非结构化数据。相对于以往便于存储的以数据库/文本为主的结构化数据,非结构化数据越来越多,包括网络日志、音频、视频、图片、地理位置信息等,这些多类型的数据对数据的处理能力提出了更高要求。



订单数据

id	用户	日期	购买商品	购买数量
1001	canglaoshi	20170710-9:10:10	面膜	2
1002	xiaozelaoshi	20170710-9:11:20	化妆品	3
1003	boduolaoshi	20170710-9:22:50	内衣	4
1004	sslaoshi	20170710-10:12:20	海狗人参丸	100



网络日志



让天下没有难学的技术

#### 4、Value (低价值密度)

价值密度的高低与数据总量的大小成反比。比如，在一天监控视频中，我们只关心宋宋老师晚上在床上健身那一分钟，如何快速对有价值数据“提纯”成为目前大数据背景下待解决的难题。



### 1.3 大数据应用场景

- 1、物流仓储：大数据分析系统助力商家精细化运营、提升销量、节约成本。  
京东物流：上午下单下午送达、下午下单次日上午送达



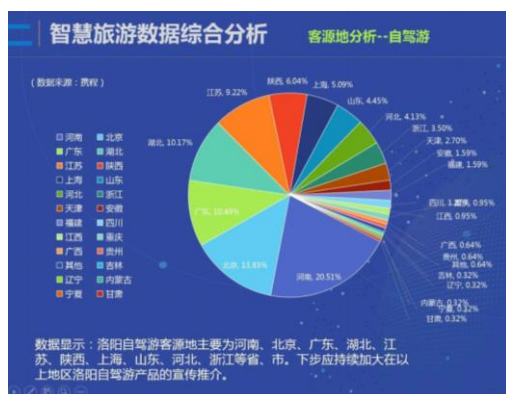
让天下没有难学的技术

2、零售：分析用户消费习惯，为用户购买商品提供方便，从而提升商品销量。

经典案例，纸尿裤+啤酒。



3、旅游：深度结合大数据能力与旅游行业需求，共建旅游产业智慧管理、智慧服务和智慧营销的未来。





### 4、商品广告推荐：给用户推荐可能喜欢的商品



我选了一种药，又推荐了8种，太棒了，么么哒！

商品已成功加入购物车！

【3万人好评 买2送1】罗博士 海狗人参丸100粒 男性保健品含淫羊藿非速效延时特...  
数量：1

[查看商品详情](#) [去购物车结算](#)

购买了该商品的用户还购买了

商品名称	价格	操作
【3万人好评 京东配送】罗博士 玛卡片玛咖100片 秘鲁进口	¥98.00	<a href="#">加入购物车</a>
罗博士 洋参温羊藿软胶囊90粒 男性保健品 非速效延时持久	¥108.00	<a href="#">加入购物车</a>
罗博士 维生素C泡腾片vc100片	¥32.00	<a href="#">加入购物车</a>
罗博士 深海牡蛎片60片 男性保健品	¥158.00	<a href="#">加入购物车</a>
罗博士 b族维生素100片复合维生素b1b2b6b12多种V8	¥38.00	<a href="#">加入购物车</a>
罗博士 成人益生面粉 复合益生元低聚果糖 2g*68g/盒	¥42.00	<a href="#">加入购物车</a>
罗博士 高浓缩玛卡60片 玛咖片黑玛咖	¥168.00	<a href="#">加入购物车</a>
罗博士 参茸红景天软胶囊100粒 男女士滋补强免疫力	¥61.80	<a href="#">加入购物车</a>

您可能还需要

1 2 3 4

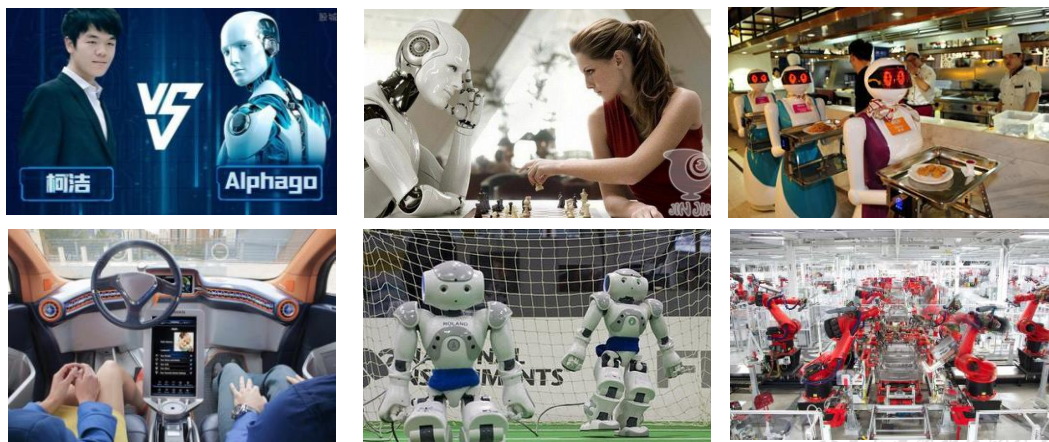
5、保险：海量数据挖掘及风险预测，助力保险行业精准营销，提升精细化定价能力。

6、金融：多维度体现用户特征，帮助金融机构推荐优质客户，防范欺诈风险。

7、房产：大数据全面助力房地产行业，打造精准投策与营销，选出更合适的地，建造更合适的楼，卖给更合适的人。



## 8、人工智能：



## 1.4 大数据发展前景

1、党的十八大提出“**实施国家大数据战略**”，国务院印发《促进大数据发展行动纲要》，大数据技术和应用处于创新突破期，国内市场需求处于爆发期，我国大数据产业面临重要的发展机遇。

2、党的十九大提出“**推动互联网、大数据、人工智能和实体经济深度融合**”。



3、国际数据公司IDC预测，到2020年，企业基于大数据计算分析平台的支出将突破5000亿美元。目前，我国大数据人才**只有46万**，未来3到5年**人才缺口达150万**之多。



人才缺口计算

$150w/5年 = 30w/年$

$30w/12月 = 2.5w/月$

自古不变的真理：先入行者吃肉，后入行者喝汤，最后到的买单！

让天下没有难学的技术

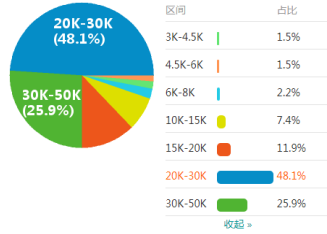
4、2017年北京大学、中国人民大学、北京邮电大学等25所高校成功申请开设大数据课程。



5、大数据属于高新技术，大牛少，升职竞争小；

让天下没有难学的技术

6、在北京大数据开发工程师的平均薪水已经到24060元（数据统计来职友集），而且目前还保持强劲的发展势头。



招聘需求地区排行 Top 10

1	北京	7851个职位
2	深圳	7544个职位
3	上海	6828个职位
4	广州	6733个职位
5	杭州	3089个职位
6	武汉	2150个职位
7	成都	1587个职位
8	合肥	1240个职位
9	南京	1203个职位
10	东莞	962个职位

让天下没有难学的技术

7、智联招聘网站上的大数据工程师薪水如下

<input type="checkbox"/> 软件工程师 (Java/大数据方向)		中金数据系统有限公司	15001-20000	北京	03-11	▼
<input type="checkbox"/> 全国各地大区大数据高级客户经理	97%	广州市冠升网络科技有限公司	20000-24999	北京	03-11	▼
<input type="checkbox"/> 大数据算法工程师	79%	北京三好互动教育科技有限公司	15001-20000	北京	03-11	▼
<input type="checkbox"/> 大数据开发工程师	66%	北京鑫信软创信息技术有限公司	15000-20000	北京	03-11	▼
<input type="checkbox"/> 高级Hadoop开发工程师 (大数据)		万科链家 (北京) 装饰有限公司	15000-30000	北京	03-11	▼
<input type="checkbox"/> 大数据分析工程师		北京百通无限网络技术有限公司	15001-20000	北京	03-11	▼
<input type="checkbox"/> 高级咨询顾问-财务、大数据、金融、信息化方向		远光软件股份有限公司北京分公司	20000-40000	北京	03-11	▼
<input type="checkbox"/> 大数据系统/算法工程师/数据工程师		北京知趣科技有限公司	15001-20000	北京	03-11	▼
<input type="checkbox"/> 大数据分析工程师		中金云金融 (北京) 大数据科技股份有限公司	12000-18000	北京-朝阳区	03-11	▼
<input type="checkbox"/> 大数据平台架构师 (java)		中金云金融 (北京) 大数据科技股份有限公司	20000-28000	北京	03-11	▼
<input type="checkbox"/> 大数据工程师-2237		完美世界 (北京) 软件有限公司	15000-25000	北京-朝阳区	03-11	▼

让天下没有难学的技术



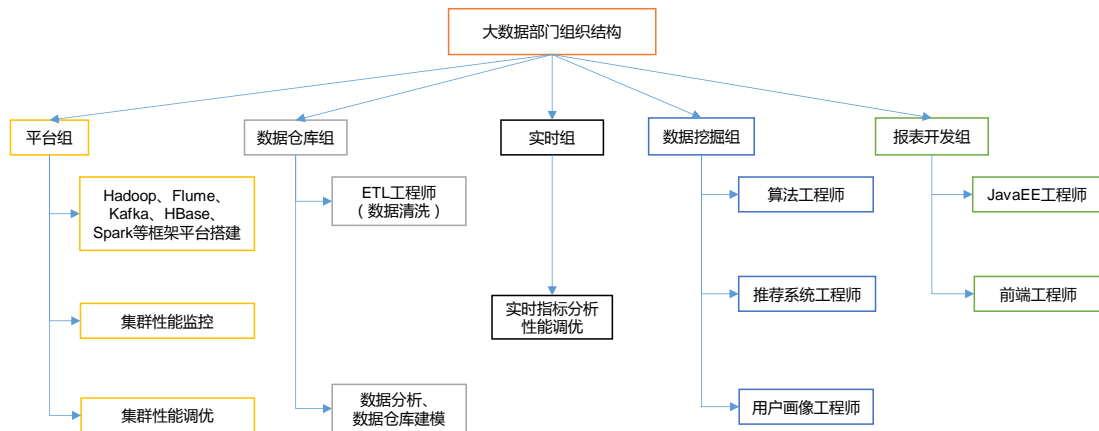
## 1.5 大数据部门业务流程分析

### 大数据部门业务流程分析



让天下没有难学的技术

## 1.6 大数据部门组织结构（重点）



让天下没有难学的技术

## 第 2 章 从 Hadoop 框架讨论大数据生态

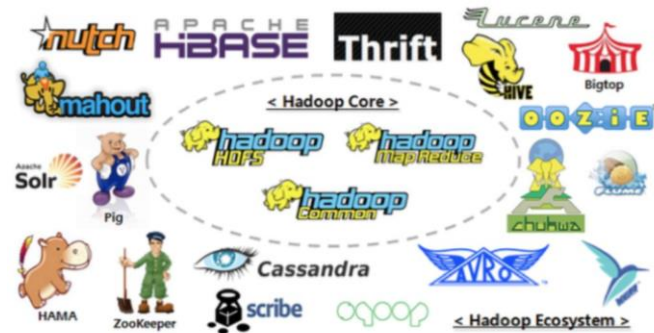
### 2.1 Hadoop 是什么



#### Hadoop是什么



- 1) Hadoop是一个由Apache基金会所开发的**分布式系统基础架构**。
- 2) 主要解决，海量数据的**存储**和海量数据的**分析计算**问题。
- 3) 广义上来说，Hadoop通常是指一个更广泛的概念——**Hadoop生态圈**。



让天下没有难学的技术

### 2.2 Hadoop 发展历史（了解）



#### Hadoop发展历史



- 1) Lucene框架是**Doug Cutting**开创的开源软件，用Java书写代码，实现与Google类似的全文搜索功能，它提供了全文检索引擎的架构，包括完整的查询引擎和索引引擎。



Hadoop创始人Doug Cutting

- 2) 2001年年底Lucene成为Apache基金会有一个子项目。
- 3) 对于海量数据的场景，Lucene面对与Google同样的困难，**存储数据困难，检索速度慢**。
- 4) 学习和模仿Google解决这些问题的办法：微型版Nutch。
- 5) 可以说Google是Hadoop的思想之源(Google在大数据方面的三篇论文)

GFS --->HDFS

Map-Reduce --->MR

Big Table --->HBase

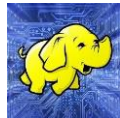
让天下没有难学的技术

6) 2003-2004年, Google公开了部分GFS和MapReduce思想的细节, 以此为基础Doug Cutting等人用了2年业余时间实现了DFS和MapReduce机制, 使Nutch性能飙升。

7) 2005 年Hadoop 作为 Lucene的子项目 Nutch的一部分正式引入Apache基金会。

8) 2006 年 3 月份, Map-Reduce和Nutch Distributed File System (NDFS) 分别被纳入到 Hadoop 项目中, Hadoop就此正式诞生, 标志着大数据时代来临。

9) 名字来源于Doug Cutting儿子的玩具大象, 如图2-20。



Hadoop的logo

让天下没有难学的技术

## 2.3 Hadoop 三大发行版本 (了解)

Hadoop 三大发行版本: Apache、Cloudera、Hortonworks。

Apache 版本最原始 (最基础) 的版本, 对于入门学习最好。

Cloudera 内部集成了很多大数据框架。对应产品 CDH。

Hortonworks 文档较好。对应产品 HDP。

### 1) Apache Hadoop

官网地址: <http://hadoop.apache.org/releases.html>

下载地址: <https://archive.apache.org/dist/hadoop/common/>

### 2) Cloudera Hadoop

官网地址: <https://www.cloudera.com/downloads/cdh/5-10-0.html>

下载地址: <http://archive-primary.cloudera.com/cdh5/cdh/5/>

(1) 2008 年成立的 Cloudera 是最早将 Hadoop 商用的公司, 为合作伙伴提供 Hadoop 的商用解决方案, 主要是包括支持、咨询服务、培训。

(2) 2009 年 Hadoop 的创始人 Doug Cutting 也加盟 Cloudera 公司。Cloudera 产品主要为 CDH, Cloudera Manager, Cloudera Support

(3) CDH 是 Cloudera 的 Hadoop 发行版, 完全开源, 比 Apache Hadoop 在兼容性, 安全性, 稳定性上有所增强。Cloudera 的标价为每年每个节点 10000 美元。

(4) Cloudera Manager 是集群的软件分发及管理监控平台, 可以在几个小时内部署好一个 Hadoop 集群, 并对集群的节点及服务进行实时监控。

### 3) Hortonworks Hadoop

官网地址: <https://hortonworks.com/products/data-center/hdp/>

下载地址: <https://hortonworks.com/downloads/#data-platform>

(1) 2011 年成立的 Hortonworks 是雅虎与硅谷风投公司 Benchmark Capital 合资组建。

(2) 公司成立之初就吸纳了大约 25 名至 30 名专门研究 Hadoop 的雅虎工程师, 上述工程师均在 2005 年开始协助雅虎开发 Hadoop, 贡献了 Hadoop 80% 的代码。

(3) Hortonworks 的主打产品是 Hortonworks Data Platform (HDP), 也同样是 100% 开源的产品, HDP 除常见的项目外还包括了 Ambari, 一款开源的安装和管理系统。

(4) Hortonworks 目前已经被 Cloudera 公司收购。

## 2.4 Hadoop 的优势 (4 高)



### Hadoop 的优势 (4 高)



1) 高可靠性: Hadoop 底层维护多个数据副本, 所以即使 Hadoop 某个计算元素或存储出现故障, 也不会导致数据的丢失。

2) 高扩展性: 在集群间分配任务数据, 可方便的扩展数以千计的节点。

3) 高效性: 在 MapReduce 的思想下, Hadoop 是并行工作的, 以加快任务处理速度。

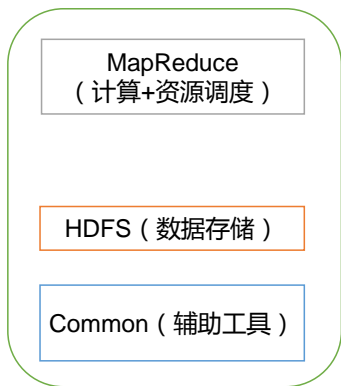
4) 高容错性: 能够自动将失败的任务重新分配。

让天下没有难学的技术

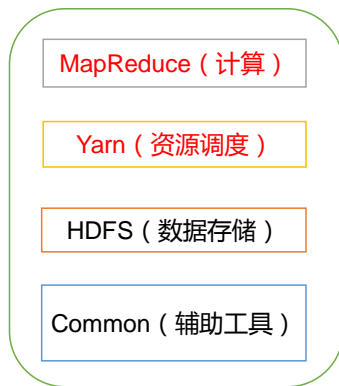
## 2.5 Hadoop 组成（面试重点）



### Hadoop1.x和Hadoop2.x区别



Hadoop1.x组成



Hadoop2.x组成

在Hadoop1.x时代，Hadoop中的MapReduce同时处理业务逻辑运算和资源的调度，耦合性较大，在Hadoop2.x时代，增加了Yarn。Yarn只负责资源的调度，MapReduce只负责运算。

让天下没有难学的技术

### 2.5.1 HDFS 架构概述



#### HDFS架构概述

1) NameNode (nn) : 存储文件的**元数据**，如**文件名**，**文件目录结构**，**文件属性**（生成时间、副本数、文件权限），以及每个文件的**块列表**和**块所在的DataNode**等。



2) DataNode(dn) : 在本地文件系统**存储文件块数据**，以及**块数据的校验和**。



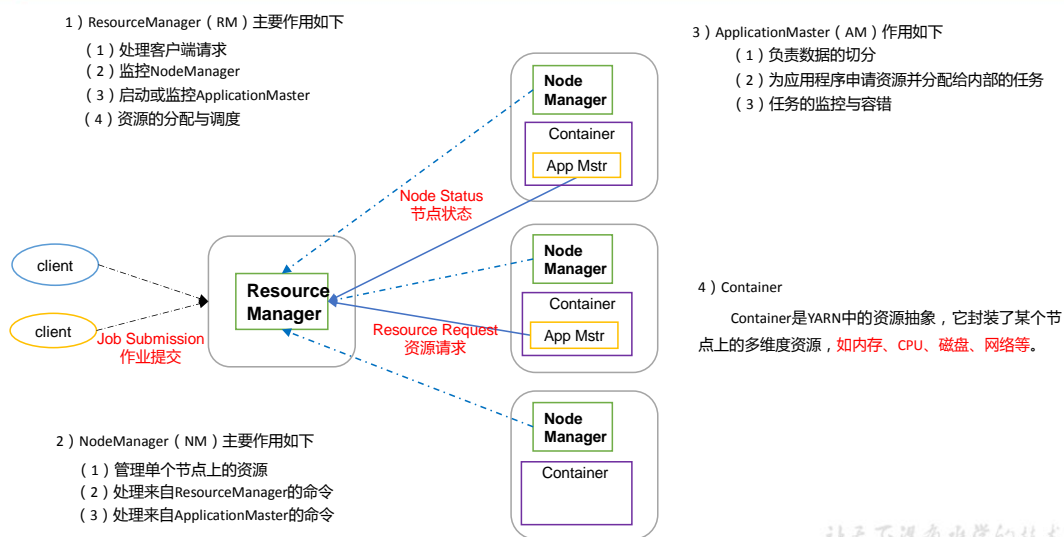
3) Secondary NameNode(2nn) : **每隔一段时间对NameNode元数据备份**。

让天下没有难学的技术



## 2.5.2 YARN 架构概述

### YARN架构



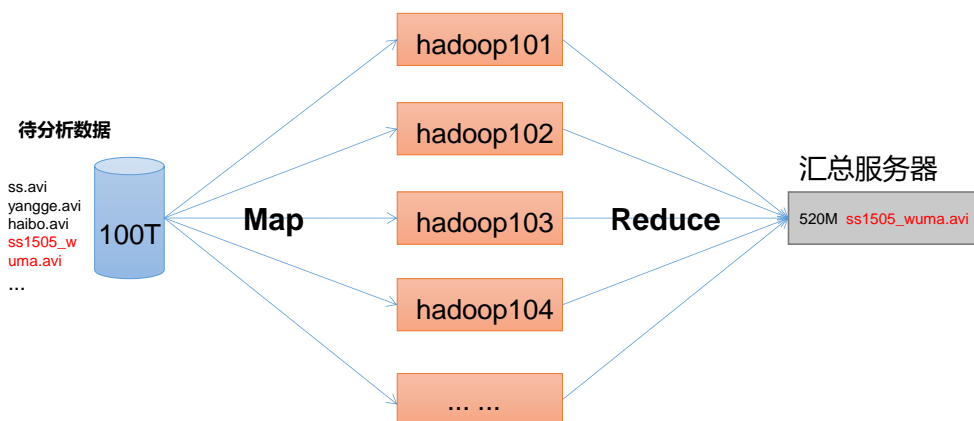
## 2.5.3 MapReduce 架构概述

MapReduce 将计算过程分为两个阶段：Map 和 Reduce

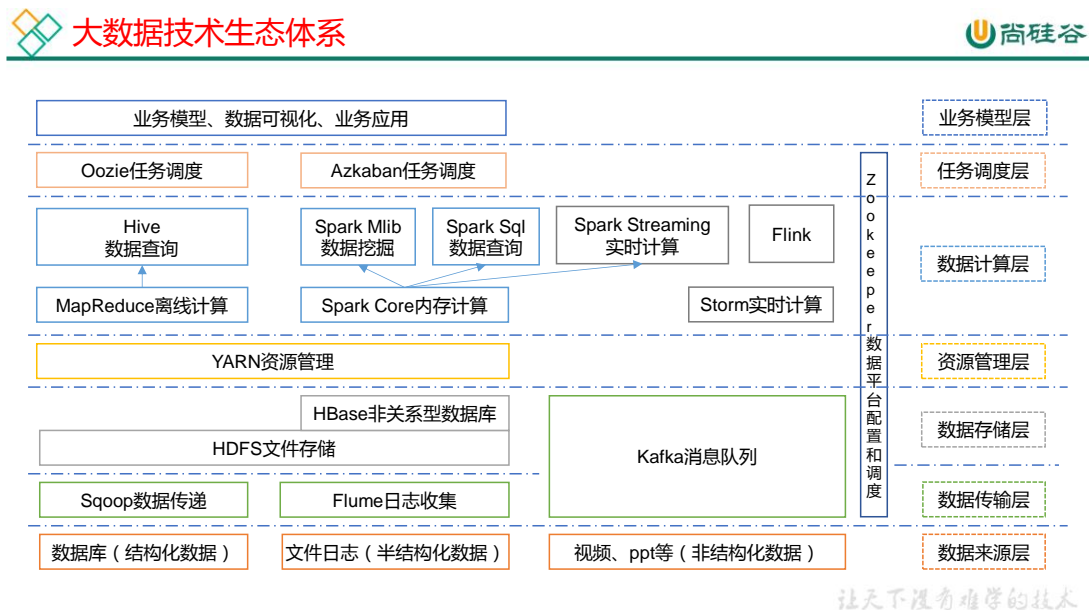
- 1) Map 阶段并行处理输入数据
- 2) Reduce 阶段对 Map 结果进行汇总



任务需求:找出宋宋老师2015年5月份的教学视频



## 2.6 大数据技术生态体系



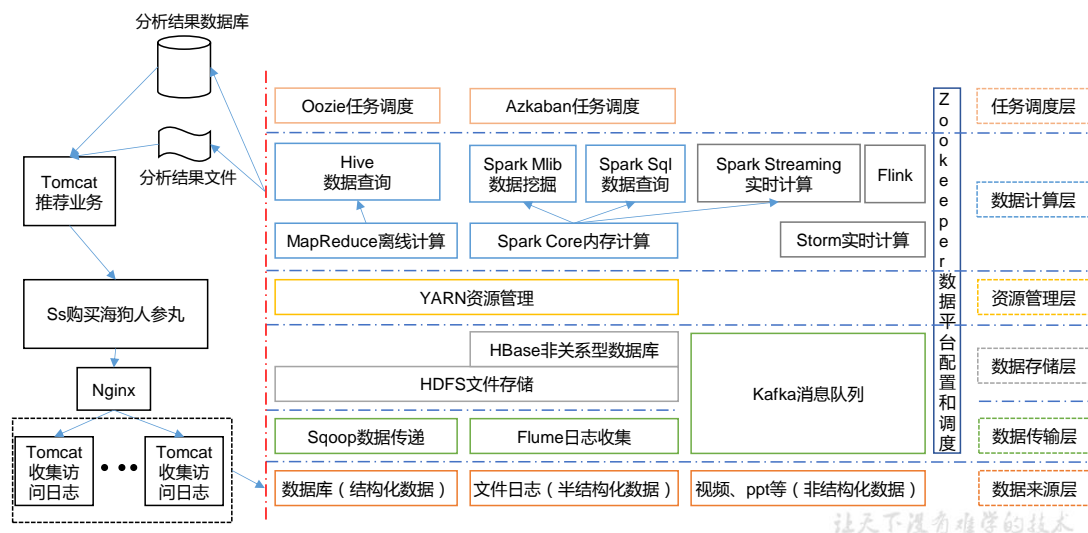
图中涉及的技术名词解释如下：

- 1) Sqoop: Sqoop 是一款开源的工具，主要用于在 Hadoop、Hive 与传统的数据库（MySQL）间进行数据的传递，可以将一个关系型数据库（例如：MySQL，Oracle 等）中的数据导进到 Hadoop 的 HDFS 中，也可以将 HDFS 的数据导进到关系型数据库中。
- 2) Flume: Flume 是一个高可用的，高可靠的，分布式的海量日志采集、聚合和传输的系统，Flume 支持在日志系统中定制各类数据发送方，用于收集数据；
- 3) Kafka: Kafka 是一种高吞吐量的分布式发布订阅消息系统；
- 4) Spark: Spark 是当前最流行的开源大数据内存计算框架。可以基于 Hadoop 上存储的大数据进行计算。
- 5) Flink: Flink 是当前最流行的开源大数据内存计算框架。用于实时计算的场景较多。
- 6) Oozie: Oozie 是一个管理 Hadoop 作业（job）的工作流程调度管理系统。
- 7) Hbase: HBase 是一个分布式的、面向列的开源数据库。HBase 不同于一般的关系数据库，它是一个适合于非结构化数据存储的数据库。
- 8) Hive: Hive 是基于 Hadoop 的一个数据仓库工具，可以将结构化的数据文件映射为一张数据库表，并提供简单的 SQL 查询功能，可以将 SQL 语句转换为 MapReduce 任务进行运行。其优点是学习成本低，可以通过类 SQL 语句快速实现简单的 MapReduce 统计，不必开发专门的 MapReduce 应用，十分适合数据仓库的统计分析。
- 9) ZooKeeper: 它是一个针对大型分布式系统的可靠协调系统，提供的功能包括：配置维护、

名字服务、分布式同步、组服务等。

## 2.7 推荐系统框架图

### 推荐系统项目框架



## 第 3 章 Hadoop 运行环境搭建（开发重点）

### 3.1 模板虚拟机环境准备

#### 1) 准备一台模板虚拟机 hadoop100，虚拟机配置要求如下：

注：本文 Linux 系统环境全部以 CentOS-7.5-x86-1804 为例说明

模板虚拟机：内存 4G，硬盘 50G，安装必要环境，为安装 hadoop 做准备

```
[root@hadoop100 ~]# yum install -y epel-release
[root@hadoop100 ~]# yum install -y psmisc nc net-tools rsync
vim lrzsz ntp libzstd openssl-static tree iotop git
```

使用 yum 安装需要虚拟机可以正常上网，yum 安装前可以先测试下虚拟机联网情况

```
[root@hadoop100 ~]# ping www.baidu.com
PING www.baidu.com (14.215.177.39) 56(84) bytes of data.
64 bytes from 14.215.177.39 (14.215.177.39): icmp_seq=1
ttl=128 time=8.60 ms
64 bytes from 14.215.177.39 (14.215.177.39): icmp_seq=2
ttl=128 time=7.72 ms
```

#### 2) 关闭防火墙，关闭防火墙开机自启

```
[root@hadoop100 ~]# systemctl stop firewalld
[root@hadoop100 ~]# systemctl disable firewalld
```

#### 3) 创建 atguigu 用户，并修改 atguigu 用户的密码

```
[root@hadoop100 ~]# useradd atguigu
[root@hadoop100 ~]# passwd atguigu
```

#### 4) 配置 atguigu 用户具有 root 权限，方便后期加 sudo 执行 root 权限的命令

```
[root@hadoop100 ~]# vim /etc/sudoers
```

修改/etc/sudoers 文件，找到下面一行（91 行），在 root 下面添加一行，如下所示：

更多 Java - 大数据 - 前端 - python 人工智能资料下载，可百度访问：尚硅谷官网

```
## Allow root to run any commands anywhere
root    ALL=(ALL)        ALL
atguigu ALL=(ALL)        NOPASSWD:ALL
```

### 5) 在/opt 目录下创建文件夹，并修改所属主和所属组

(1) 在/opt 目录下创建 module、software 文件夹

```
[root@hadoop100 ~]# mkdir /opt/module
[root@hadoop100 ~]# mkdir /opt/software
```

(2) 修改 module、software 文件夹的所有者和所属组均为 atguigu 用户

```
[root@hadoop100 ~]# chown atguigu:atguigu /opt/module
[root@hadoop100 ~]# chown atguigu:atguigu /opt/software
```

(3) 查看 module、software 文件夹的所有者和所属组

```
[root@hadoop100 ~]# cd /opt/
[root@hadoop100 opt]# ll
总用量 12
drwxr-xr-x. 2 atguigu atguigu 4096 5月 28 17:18 module
drwxr-xr-x. 2 root    root    4096 9月 7 2017 rh
drwxr-xr-x. 2 atguigu atguigu 4096 5月 28 17:18 software
```

### 6) 卸载虚拟机自带的 open JDK

```
[root@hadoop100 ~]# rpm -qa | grep -i java | xargs -n1 rpm -e --nodeps
```

### 7) 重启虚拟机

```
[root@hadoop100 ~]# reboot
```

## 3.2 克隆虚拟机

### 1) 利用模板机 hadoop100，克隆三台虚拟机：hadoop102 hadoop103 hadoop104

### 2) 修改克隆机 IP，以下以 hadoop102 举例说明

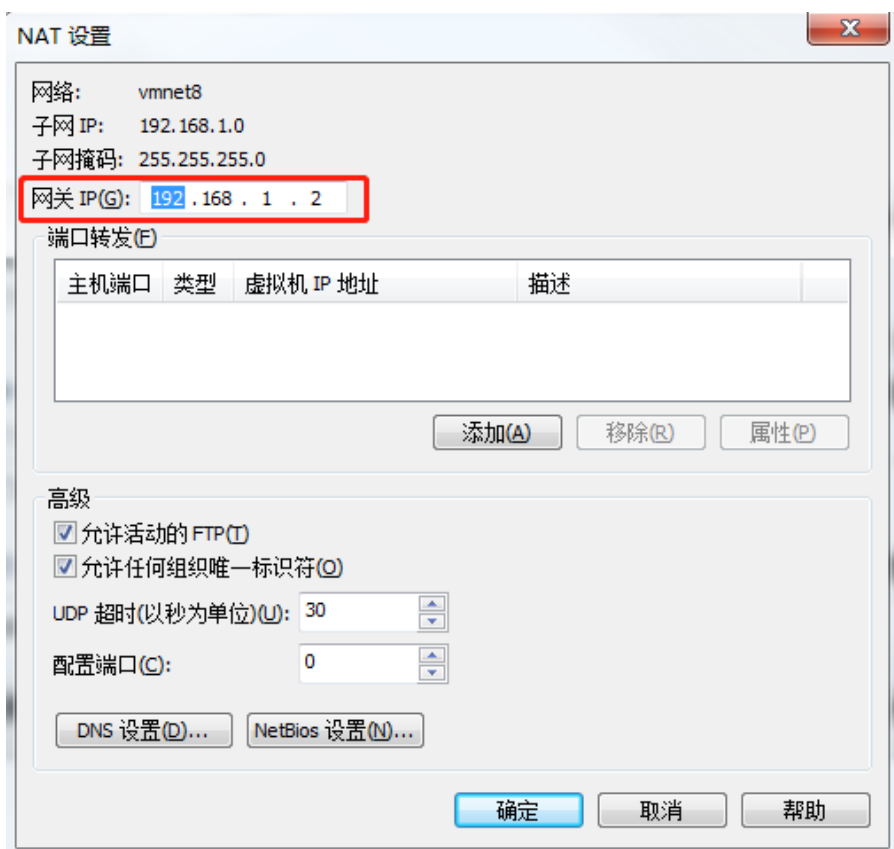
(1) 修改克隆虚拟机的静态 IP

```
[root@hadoop100 ~]# vim /etc/sysconfig/network-scripts/ifcfg-ens33
```

改成

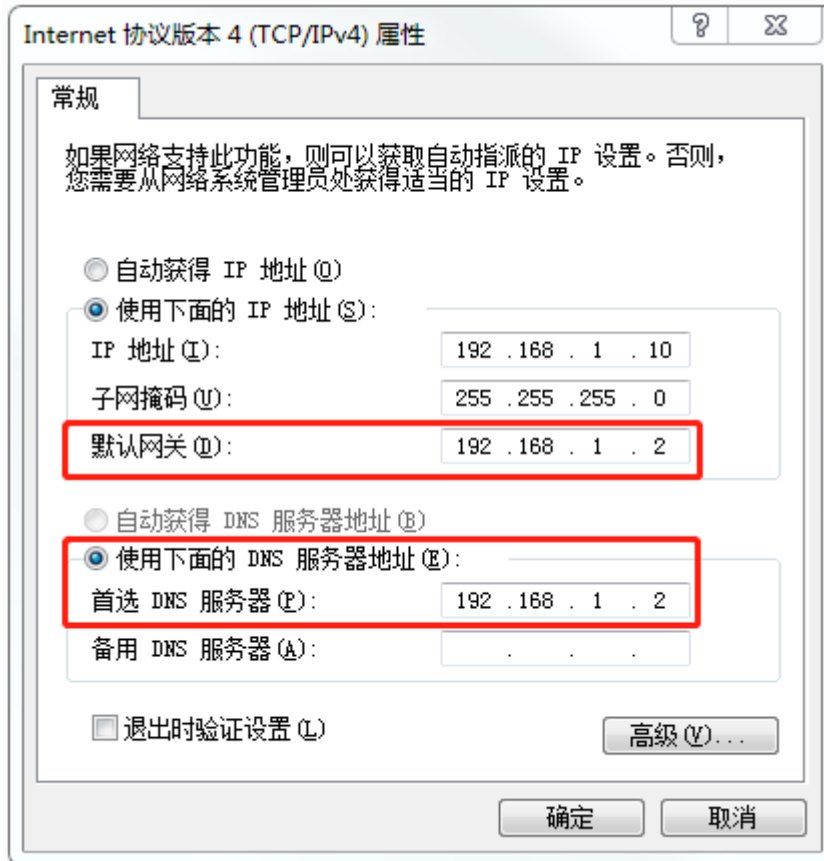
```
DEVICE=ens33
TYPE=Ethernet
ONBOOT=yes
BOOTPROTO=static
NAME="ens33"
IPADDR=192.168.1.102
PREFIX=24
GATEWAY=192.168.1.2
DNS1=192.168.1.2
```

(2) 查看 Linux 虚拟机的虚拟网络编辑器，编辑->虚拟网络编辑器->VMnet8



(3) 查看 Windows 系统适配器 VMware Network Adapter VMnet8 的 IP 地址





(4) 保证 Linux 系统 ifcfg-ens33 文件中 IP 地址、虚拟网络编辑器地址和 Windows 系统 VM8 网络 IP 地址相同。

### 3) 修改克隆机主机名，以下以 **hadoop102** 举例说明

(1) 修改主机名称，两种方法二选一

```
[root@hadoop100 ~]# hostnamectl --static set-hostname hadoop102
```

或者修改/etc/hostname 文件

```
[root@hadoop100 ~]# vim /etc/hostname
hadoop102
```

(2) 配置 linux 克隆机主机名称映射 hosts 文件，打开/etc/hosts

```
[root@hadoop100 ~]# vim /etc/hosts
```

添加如下内容

```
192.168.1.100 hadoop100
192.168.1.101 hadoop101
192.168.1.102 hadoop102
192.168.1.103 hadoop103
192.168.1.104 hadoop104
192.168.1.105 hadoop105
192.168.1.106 hadoop106
192.168.1.107 hadoop107
192.168.1.108 hadoop108
```

### 4) 重启克隆机 **hadoop102**

```
[root@hadoop100 ~]# reboot
```

## 5) 修改 windows 的主机映射文件 (hosts 文件)

(1) 如果操作系统是 window7, 可以直接修改

(a) 进入 C:\Windows\System32\drivers\etc 路径

(b) 打开 hosts 文件并添加如下内容, 然后保存

```
192.168.1.100 hadoop100
192.168.1.101 hadoop101
192.168.1.102 hadoop102
192.168.1.103 hadoop103
192.168.1.104 hadoop104
192.168.1.105 hadoop105
192.168.1.106 hadoop106
192.168.1.107 hadoop107
192.168.1.108 hadoop108
```

(2) 如果操作系统是 window10, 先拷贝出来, 修改保存以后, 再覆盖即可

(a) 进入 C:\Windows\System32\drivers\etc 路径

(b) 拷贝 hosts 文件到桌面

(c) 打开桌面 hosts 文件并添加如下内容

```
192.168.1.100 hadoop100
192.168.1.101 hadoop101
192.168.1.102 hadoop102
192.168.1.103 hadoop103
192.168.1.104 hadoop104
192.168.1.105 hadoop105
192.168.1.106 hadoop106
192.168.1.107 hadoop107
192.168.1.108 hadoop108
```

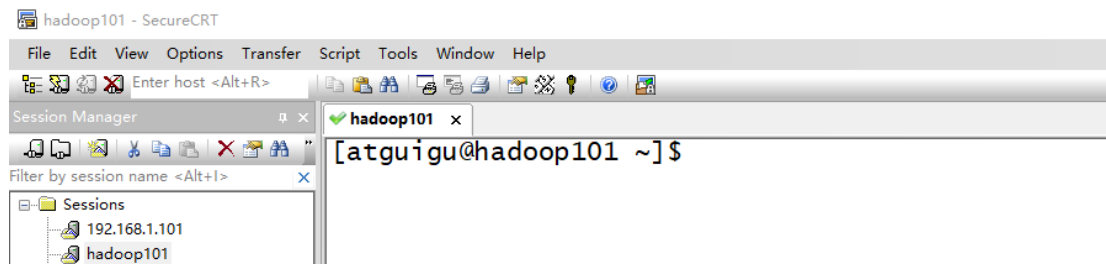
(d) 将桌面 hosts 文件覆盖 C:\Windows\System32\drivers\etc 路径 hosts 文件

## 3.3 在 hadoop102 安装 JDK

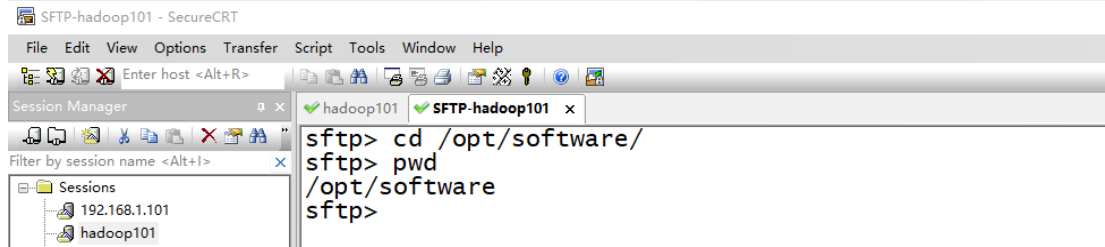
### 1) 卸载现有 JDK

```
[atguigu@hadoop102 ~]$ rpm -qa | grep -i java | xargs -n1 sudo
rpm -e --nodeps
```

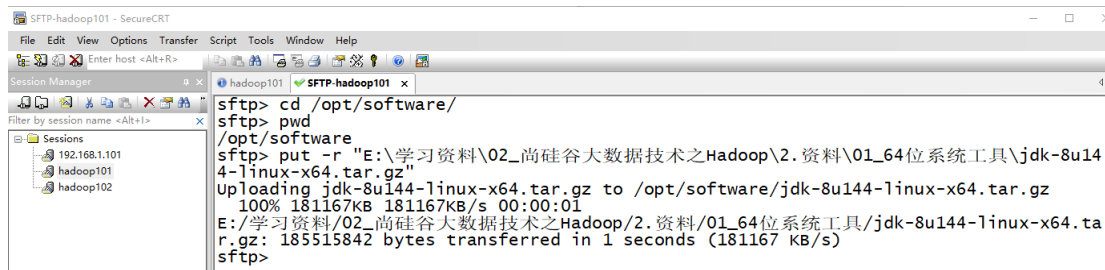
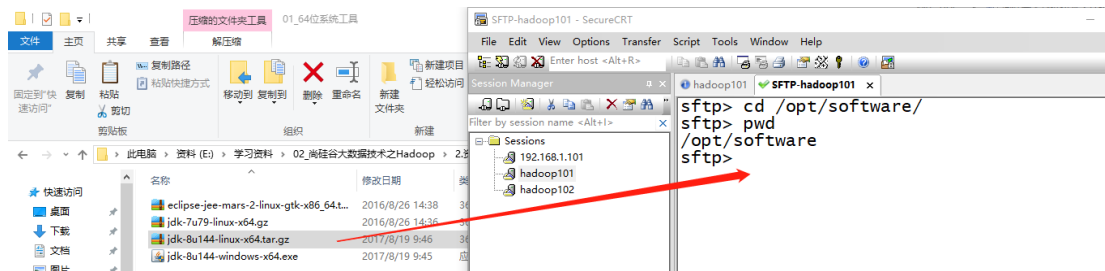
### 2) 用 SecureCRT 工具将 JDK 导入到 opt 目录下面的 software 文件夹下面



### 3) “alt+p”进入 sftp 模式



## 4) 选择 jdk1.8 拖入工具



## 5) 在 Linux 系统下的 opt 目录中查看软件包是否导入成功

```
[atguigu@hadoop102 ~]$ ls /opt/software/
```

看到如下结果:

```
hadoop-3.1.3.tar.gz  jdk-8u212-linux-x64.tar.gz
```

## 6) 解压 JDK 到/opt/module 目录下

```
[atguigu@hadoop102 software]$ tar -zxvf jdk-8u212-linux-x64.tar.gz -C /opt/module/
```

## 7) 配置 JDK 环境变量

(1) 新建/etc/profile.d/my\_env.sh 文件

```
[atguigu@hadoop102 ~]$ sudo vim /etc/profile.d/my_env.sh
```

添加如下内容

```
#JAVA_HOME
export JAVA_HOME=/opt/module/jdk1.8.0_212
export PATH=$PATH:$JAVA_HOME/bin
```

(2) 保存后退出

```
:wq
```

(3) source 一下/etc/profile 文件, 让新的环境变量 PATH 生效

```
[atguigu@hadoop102 ~]$ source /etc/profile
```

## 8) 测试 JDK 是否安装成功

```
[atguigu@hadoop102 ~]$ java -version
```

如果能看到以下结果，则代表 Java 安装成功。

```
java version "1.8.0_212"
```

注意：重启（如果 java -version 可以用就不用重启）

```
[atguigu@hadoop102 ~]$ sudo reboot
```

### 3.4 在 hadoop102 安装 Hadoop

Hadoop 下载地址：<https://archive.apache.org/dist/hadoop/common/hadoop-3.1.3/>

1) 用 SecureCRT 工具将 hadoop-3.1.3.tar.gz 导入到 opt 目录下面的 software 文件夹下面

切换到 sftp 连接页面，选择 Linux 下编译的 hadoop jar 包拖入，如图 2-32 所示

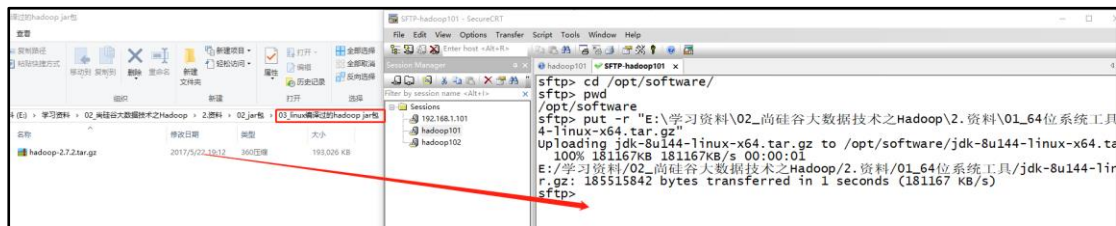


图 2-32 拖入 hadoop 的 tar 包

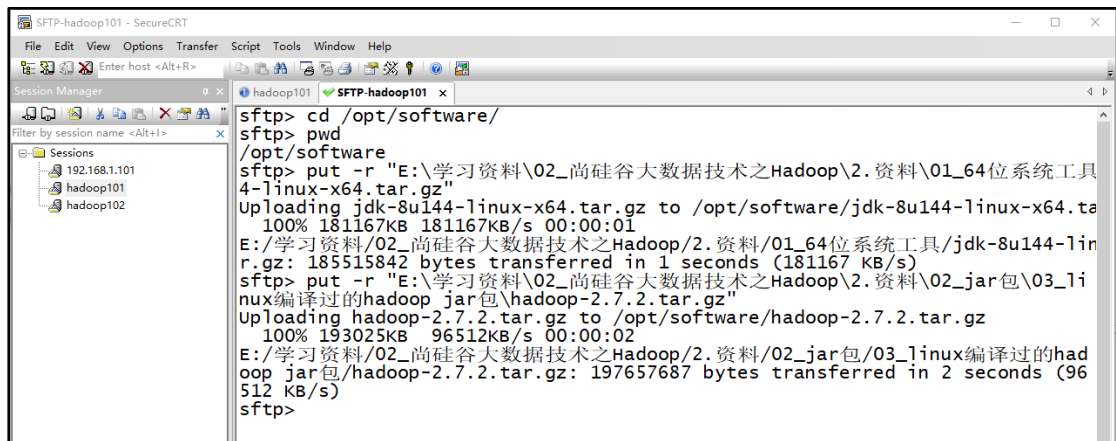


图 2-33 拖入 Hadoop 的 tar 包成功

2) 进入到 Hadoop 安装包路径下

```
[atguigu@hadoop102 ~]$ cd /opt/software/
```

3) 解压安装文件到/opt/module 下面

```
[atguigu@hadoop102 software]$ tar -zxvf hadoop-3.1.3.tar.gz -C /opt/module/
```

4) 查看是否解压成功

```
[atguigu@hadoop102 software]$ ls /opt/module/  
hadoop-3.1.3
```

5) 将 Hadoop 添加到环境变量

(1) 获取 Hadoop 安装路径

```
[atguigu@hadoop102 hadoop-3.1.3]$ pwd  
/opt/module/hadoop-3.1.3
```

(2) 打开/etc/profile.d/my\_env.sh 文件

```
sudo vim /etc/profile.d/my_env.sh
```

在 my\_env.sh 文件末尾添加如下内容: (shift+g)

```
#HADOOP_HOME
export HADOOP_HOME=/opt/module/hadoop-3.1.3
export PATH=$PATH:$HADOOP_HOME/bin
export PATH=$PATH:$HADOOP_HOME/sbin
```

(3) 保存后退出

```
:wq
```

(4) 让修改后的文件生效

```
[atguigu@hadoop102 hadoop-3.1.3]$ source /etc/profile
```

## 6) 测试是否安装成功

```
[atguigu@hadoop102 hadoop-3.1.3]$ hadoop version
Hadoop 3.1.3
```

## 7) 重启(如果 Hadoop 命令不能用再重启)

```
[atguigu@hadoop102 hadoop-3.1.3]$ sync
[atguigu@hadoop102 hadoop-3.1.3]$ sudo reboot
```

# 3.5 Hadoop 目录结构

## 1) 查看 Hadoop 目录结构

```
[atguigu@hadoop102 hadoop-3.1.3]$ ll
总用量 52
drwxr-xr-x. 2 atguigu atguigu 4096 5月 22 2017 bin
drwxr-xr-x. 3 atguigu atguigu 4096 5月 22 2017 etc
drwxr-xr-x. 2 atguigu atguigu 4096 5月 22 2017 include
drwxr-xr-x. 3 atguigu atguigu 4096 5月 22 2017 lib
drwxr-xr-x. 2 atguigu atguigu 4096 5月 22 2017 libexec
-rw-r--r--. 1 atguigu atguigu 15429 5月 22 2017 LICENSE.txt
-rw-r--r--. 1 atguigu atguigu 101 5月 22 2017 NOTICE.txt
-rw-r--r--. 1 atguigu atguigu 1366 5月 22 2017 README.txt
drwxr-xr-x. 2 atguigu atguigu 4096 5月 22 2017 sbin
drwxr-xr-x. 4 atguigu atguigu 4096 5月 22 2017 share
```

## 2) 重要目录

- (1) bin 目录: 存放对 Hadoop 相关服务 (HDFS,YARN) 进行操作的脚本
- (2) etc 目录: Hadoop 的配置文件目录, 存放 Hadoop 的配置文件
- (3) lib 目录: 存放 Hadoop 的本地库 (对数据进行压缩解压缩功能)
- (4) sbin 目录: 存放启动或停止 Hadoop 相关服务的脚本
- (5) share 目录: 存放 Hadoop 的依赖 jar 包、文档、和官方案例



## 第 4 章 Hadoop 运行模式

Hadoop 运行模式包括：本地模式、伪分布式模式以及完全分布式模式。

Hadoop 官方网站：<http://hadoop.apache.org/>

### 4.1 本地运行模式（官方 wordcount）

- 1) 创建在 **hadoop-3.1.3** 文件下面创建一个 **wcinput** 文件夹

```
[atguigu@hadoop102 hadoop-3.1.3]$ mkdir wcinput
```

- 2) 在 **wcinput** 文件下创建一个 **word.txt** 文件

```
[atguigu@hadoop102 hadoop-3.1.3]$ cd wcinput
```

- 3) 编辑 **word.txt** 文件

```
[atguigu@hadoop102 wcinput]$ vim word.txt
```

在文件中输入如下内容

```
hadoop yarn
hadoop mapreduce
atguigu
atguigu
```

保存退出：：wq

- 4) 回到 **Hadoop** 目录/opt/module/hadoop-3.1.3

- 5) 执行程序

```
[atguigu@hadoop102 hadoop-3.1.3]$ hadoop jar
share/hadoop/mapreduce/hadoop-mapreduce-examples-3.1.3.jar
wordcount wcinput wcoutput
```

- 6) 查看结果

```
[atguigu@hadoop102 hadoop-3.1.3]$ cat wcoutput/part-r-00000
```

看到如下结果：

```
atguigu 2
hadoop 2
mapreduce 1
yarn 1
```

### 4.2 完全分布式运行模式（开发重点）

分析：

- 1) 准备 3 台客户机（**关闭防火墙、静态 ip、主机名称**）
- 2) 安装 JDK
- 3) 配置环境变量
- 4) 安装 Hadoop
- 5) 配置环境变量

- 6) 配置集群
- 7) 单点启动
- 8) 配置 ssh
- 9) 群起并测试集群

### 4.2.1 虚拟机准备

详见 3.1,3.2 两章。

### 4.2.2 编写集群分发脚本 xsync

#### 1) scp (secure copy) 安全拷贝

(1) scp 定义:

scp 可以实现服务器与服务器之间的数据拷贝。(from server1 to server2)

(2) 基本语法

scp	-r	\$pdir/\$fname	\$user@hadoop\$host:\$pdir/\$fname
命令	递归	要拷贝的文件路径/名称	目的用户@主机:目的路径/名称

(3) 案例实操

前提: 在 hadoop102 hadoop103 hadoop104 都已经创建好的 /opt/module

/opt/software 两个目录, 并且已经把这两个目录修改为 atguigu:atguigu

sudo chown atguigu:atguigu -R /opt/module

(a) 在 hadoop102 上, 将 hadoop102 中 /opt/module/jdk1.8.0\_212 目录拷贝到 hadoop103 上。

```
[atguigu@hadoop102 ~]$ scp -r /opt/module/jdk1.8.0_212  
atguigu@hadoop103:/opt/module
```

(b) 在 hadoop103 上, 将 hadoop102 中 /opt/module/hadoop-3.1.3 目录拷贝到 hadoop103 上。

```
[atguigu@hadoop103 ~]$ scp -r  
atguigu@hadoop102:/opt/module/hadoop-3.1.3 /opt/module/
```

(c) 在 hadoop103 上操作, 将 hadoop102 中 /opt/module 目录下所有目录拷贝到 hadoop104 上。

```
[atguigu@hadoop103 ~]$ scp -r  
atguigu@hadoop102:/opt/module/*  
atguigu@hadoop104:/opt/module
```

#### 2) rsync 远程同步工具

rsync 主要用于备份和镜像。具有速度快、避免复制相同内容和支持符号链接的优点。

rsync 和 scp 区别: 用 rsync 做文件的复制要比 scp 的速度快, rsync 只对差异文件做更

新。scp 是把所有文件都复制过去。

## (1) 基本语法

rsync      -av              \$pdir/\$fname                      \$user@hadoop\$host:\$pdir/\$fname  
命令    选项参数    要拷贝的文件路径/名称      目的用户@主机:目的路径/名称

选项参数说明

选项	功能
-a	归档拷贝
-v	显示复制过程

## (2) 案例实操

(a) 把 hadoop102 机器上的 /opt/software 目录同步到 hadoop103 服务器的 /opt/software 目录下

```
[atguigu@hadoop102 opt]$ rsync -av /opt/software/*
atguigu@hadoop103:/opt/software
```

## 3) xsync 集群分发脚本

(1) 需求：循环复制文件到所有节点的相同目录下

(2) 需求分析：

(a) rsync 命令原始拷贝：

```
rsync -av /opt/module root@hadoop103:/opt/
```

(b) 期望脚本：

xsync 要同步的文件名称

(c) 说明：在 /home/atguigu/bin 这个目录下存放的脚本，atguigu 用户可以在系统任何地方直接执行。

## (3) 脚本实现

(a) 在 /home/atguigu/bin 目录下创建 xsync 文件

```
[atguigu@hadoop102 opt]$ cd /home/atguigu
[atguigu@hadoop102 ~]$ mkdir bin
[atguigu@hadoop102 ~]$ cd bin
[atguigu@hadoop102 bin]$ vim xsync
```

在该文件中编写如下代码

```
#!/bin/bash
#1. 判断参数个数
if [ $# -lt 1 ]
then
    echo Not Enough Argument!
    exit;
fi
#2. 遍历集群所有机器
for host in hadoop102 hadoop103 hadoop104
do
```

```
echo ===== $host =====
#3. 遍历所有目录，挨个发送
for file in $@
do
    #4. 判断文件是否存在
    if [ -e $file ]
    then
        #5. 获取父目录
        pdir=$(cd -P $(dirname $file); pwd)
        #6. 获取当前文件的名称
        fname=$(basename $file)
        ssh $host "mkdir -p $pdir"
        rsync -av $pdir/$fname $host:$pdir
    else
        echo $file does not exists!
    fi
done
done
```

(b) 修改脚本 xsync 具有执行权限

```
[atguigu@hadoop102 bin]$ chmod +x xsync
```

(c) 将脚本复制到/bin 中，以便全局调用

```
[atguigu@hadoop102 bin]$ sudo cp xsync /bin/
```

(d) 测试脚本

```
[atguigu@hadoop102 ~]$ xsync /home/atguigu/bin
[atguigu@hadoop102 bin]$ sudo xsync /bin/xsync
```

## 4.2.3 SSH 无密登录配置

### 1) 配置 ssh

(1) 基本语法

ssh 另一台电脑的 ip 地址

(2) ssh 连接时出现 Host key verification failed 的解决方法

```
[atguigu@hadoop102 ~]$ ssh hadoop103
```

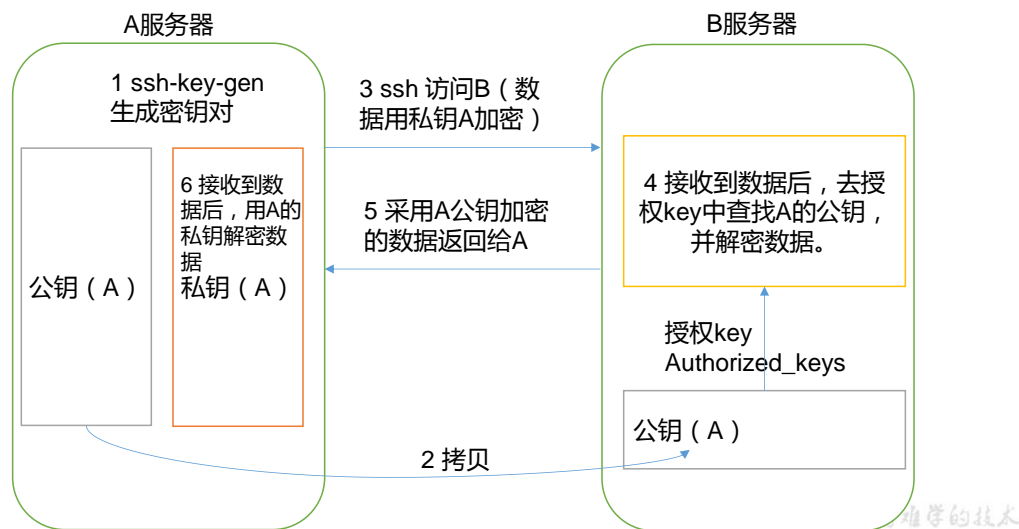
出现:

```
The authenticity of host '192.168.1.103 (192.168.1.103)' can't
be established.
RSA key fingerprint is
cf:1e:de:d7:d0:4c:2d:98:60:b4:fd:ae:b1:2d:ad:06.
Are you sure you want to continue connecting (yes/no)?
```

(3) 解决方案如下: 直接输入 yes

### 2) 无密钥配置

(1) 免密登录原理



(2) 生成公钥和私钥:

```
[atguigu@hadoop102 .ssh]$ ssh-keygen -t rsa
```

然后敲 (三个回车)，就会生成两个文件 id\_rsa (私钥)、id\_rsa.pub (公钥)

(3) 将公钥拷贝到要免密登录的目标机器上

```
[atguigu@hadoop102 .ssh]$ ssh-copy-id hadoop102
[atguigu@hadoop102 .ssh]$ ssh-copy-id hadoop103
[atguigu@hadoop102 .ssh]$ ssh-copy-id hadoop104
```

注意:

还需要在 hadoop103 上采用 atguigu 账号配置一下无密登录到 hadoop102、hadoop103、hadoop104 服务器上。

还需要在 hadoop104 上采用 atguigu 账号配置一下无密登录到 hadoop102、hadoop103、hadoop104 服务器上。

还需要在 hadoop102 上采用 root 账号，配置一下无密登录到 hadoop102、hadoop103、hadoop104;

### 3) .ssh 文件夹下 (~/.ssh) 的文件功能解释

known_hosts	记录 ssh 访问过计算机的公钥(public key)
id_rsa	生成的私钥
id_rsa.pub	生成的公钥
authorized_keys	存放授权过的无密登录服务器公钥

## 4.2.4 集群配置

### 1) 集群部署规划

更多 [Java](#) - [大数据](#) - [前端](#) - [python](#) 人工智能资料下载，可百度访问：尚硅谷官网



注意：NameNode 和 SecondaryNameNode 不要安装在同一台服务器

注意：ResourceManager 也很消耗内存，不要和 NameNode、SecondaryNameNode 配置在同一台机器上。

	hadoop102	hadoop103	hadoop104
HDFS	NameNode DataNode	DataNode	SecondaryNameNode DataNode
YARN	NodeManager	ResourceManager NodeManager	NodeManager

## 2) 配置文件说明

Hadoop 配置文件分两类：默认配置文件和自定义配置文件，只有用户想修改某一默认配置值时，才需要修改自定义配置文件，更改相应属性值。

(1) 默认配置文件：

要获取的默认文件	文件存放在 Hadoop 的 jar 包中的位置
[core-default.xml]	hadoop-common-3.1.3.jar/ core-default.xml
[hdfs-default.xml]	hadoop-hdfs-3.1.3.jar/ hdfs-default.xml
[yarn-default.xml]	hadoop-yarn-common-3.1.3.jar/ yarn-default.xml
[mapred-default.xml]	hadoop-mapreduce-client-core-3.1.3.jar/ mapred-default.xml

(2) 自定义配置文件：

**core-site.xml**、**hdfs-site.xml**、**yarn-site.xml**、**mapred-site.xml** 四个配置文件存放在 \$HADOOP\_HOME/etc/hadoop 这个路径上，用户可以根据项目需求重新进行修改配置。

(3) 常用端口号说明

Daemon	App	Hadoop2	Hadoop3
NameNode Port	Hadoop HDFS NameNode	8020 / 9000	9820
	Hadoop HDFS NameNode HTTP UI	50070	9870
Secondary NameNode Port	Secondary NameNode	50091	9869
	Secondary NameNode HTTP UI	50090	9868
DataNode Port	Hadoop HDFS DataNode IPC	50020	9867

	Hadoop HDFS DataNode	50010	9866
	Hadoop HDFS DataNode HTTP UI	50075	9864

### 3) 配置集群

#### (1) 核心配置文件

配置 core-site.xml

```
[atguigu@hadoop102 ~]$ cd $HADOOP_HOME/etc/hadoop
[atguigu@hadoop102 hadoop]$ vim core-site.xml
```

文件内容如下:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<configuration>
  <!-- 指定 NameNode 的地址 -->
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://hadoop102:9820</value>
  </property>
  <!-- 指定 hadoop 数据的存储目录 -->
  <property>
    <name>hadoop.tmp.dir</name>
    <value>/opt/module/hadoop-3.1.3/data</value>
  </property>

  <!-- 配置 HDFS 网页登录使用的静态用户为 atguigu -->
  <property>
    <name>hadoop.http.staticuser.user</name>
    <value>atguigu</value>
  </property>

  <!-- 配置该 atguigu(superUser) 允许通过代理访问的主机节点 -->
  <property>
    <name>hadoop.proxyuser.atguigu.hosts</name>
    <value>*</value>
  </property>
  <!-- 配置该 atguigu(superUser) 允许通过代理用户所属组 -->
  <property>
    <name>hadoop.proxyuser.atguigu.groups</name>
    <value>*</value>
  </property>
  <!-- 配置该 atguigu(superUser) 允许通过代理的用户 -->
  <property>
    <name>hadoop.proxyuser.atguigu.groups</name>
    <value>*</value>
  </property>
</configuration>
```

#### (2) HDFS 配置文件

配置 hdfs-site.xml

```
[atguigu@hadoop102 hadoop]$ vim hdfs-site.xml
```

文件内容如下:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<configuration>
  <!-- nn web 端访问地址-->
  <property>
    <name>dfs.namenode.http-address</name>
    <value>hadoop102:9870</value>
  </property>
  <!-- 2nn web 端访问地址-->
  <property>
    <name>dfs.namenode.secondary.http-address</name>
    <value>hadoop104:9868</value>
  </property>
</configuration>
```

### (3) YARN 配置文件

配置 yarn-site.xml

```
[atguigu@hadoop102 hadoop]$ vim yarn-site.xml
```

文件内容如下:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<configuration>
  <!-- 指定 MR 走 shuffle -->
  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
  </property>
  <!-- 指定 ResourceManager 的地址-->
  <property>
    <name>yarn.resourcemanager.hostname</name>
    <value>hadoop103</value>
  </property>
  <!-- 环境变量的继承 -->
  <property>
    <name>yarn.nodemanager.env-whitelist</name>
    <value>JAVA_HOME,HADOOP_COMMON_HOME,HADOOP_HDFS_HOME,HADOOP_CONF_DIR,CLASSPATH_PREPEND_DISTCACHE,HADOOP_YARN_HOME,HADOOP_MAPRED_HOME</value>
  </property>
  <!-- yarn 容器允许分配的最大最小内存 -->
  <property>
    <name>yarn.scheduler.minimum-allocation-mb</name>
    <value>512</value>
  </property>
  <property>
    <name>yarn.scheduler.maximum-allocation-mb</name>
    <value>4096</value>
  </property>
  <!-- yarn 容器允许管理的物理内存大小 -->
```

```
<property>
  <name>yarn.nodemanager.resource.memory-mb</name>
  <value>4096</value>
</property>
<!-- 关闭 yarn 对物理内存和虚拟内存的限制检查 -->
<property>
  <name>yarn.nodemanager.pmem-check-enabled</name>
  <value>>false</value>
</property>
<property>
  <name>yarn.nodemanager.vmem-check-enabled</name>
  <value>>false</value>
</property>
</configuration>
```

#### (4) MapReduce 配置文件

配置 mapred-site.xml

```
[atguigu@hadoop102 hadoop]$ vim mapred-site.xml
```

文件内容如下:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<configuration>
  <!-- 指定 MapReduce 程序运行在 Yarn 上 -->
  <property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
  </property>
</configuration>
```

#### 4) 在集群上分发配置好的 Hadoop 配置文件

```
[atguigu@hadoop102 hadoop]$ xsync /opt/module/hadoop-3.1.3/etc/hadoop/
```

#### 5) 去 103 和 104 上查看文件分发情况

```
[atguigu@hadoop103 ~]$ cat /opt/module/hadoop-3.1.3/etc/hadoop/core-site.xml
[atguigu@hadoop104 ~]$ cat /opt/module/hadoop-3.1.3/etc/hadoop/core-site.xml
```

### 4.2.5 群起集群

#### 1) 配置 workers

```
[atguigu@hadoop102 hadoop]$ vim /opt/module/hadoop-3.1.3/etc/hadoop/workers
```

在该文件中增加如下内容:

```
hadoop102
hadoop103
hadoop104
```

**注意:** 该文件中添加的内容结尾不允许有空格, 文件中不允许有空行。

同步所有节点配置文件

```
[atguigu@hadoop102 hadoop]$ xsync /opt/module/hadoop-3.1.3/etc
```

## 2) 启动集群

(1) **如果集群是第一次启动**，需要在 hadoop102 节点格式化 NameNode（注意格式化 NameNode，会产生新的集群 id，导致 NameNode 和 DataNode 的集群 id 不一致，集群找不到已往数据。如果集群在运行过程中报错，需要重新格式化 NameNode 的话，一定要先停止 namenode 和 datanode 进程，并且要删除所有机器的 data 和 logs 目录，然后再进行格式化。）

```
[atguigu@hadoop102 ~]$ hdfs namenode -format
```

(2) 启动 HDFS

```
[atguigu@hadoop102 hadoop-3.1.3]$ sbin/start-dfs.sh
```

(3) 在配置了 **ResourceManager** 的节点 (**hadoop103**) 启动 YARN

```
[atguigu@hadoop103 hadoop-3.1.3]$ sbin/start-yarn.sh
```

(4) Web 端查看 HDFS 的 NameNode

(a) 浏览器中输入: <http://hadoop102:9870>

(b) 查看 HDFS 上存储的数据信息

(5) Web 端查看 YARN 的 ResourceManager

(a) 浏览器中输入: <http://hadoop103:8088>

(b) 查看 YARN 上运行的 Job 信息

## 3) 集群基本测试

(1) 上传文件到集群

上传小文件

```
[atguigu@hadoop102 ~]$ hadoop fs -mkdir /input
[atguigu@hadoop102 ~]$ hadoop fs -put $HADOOP_HOME/wcinput/word.txt /input
```

上传大文件

```
[atguigu@hadoop102 ~]$ hadoop fs -put /opt/software/jdk-8u212-linux-x64.tar.gz /
```

(2) 上传文件后查看文件存放在什么位置

(a) 查看 HDFS 文件存储路径

```
[atguigu@hadoop102 subdir0]$ pwd
/opt/module/hadoop-3.1.3/data/dfs/data/current/BP-938951106-192.168.10.107-1495462844069/current/finalized/subdir0/subdir0
```

(b) 查看 HDFS 在磁盘存储文件内容

```
[atguigu@hadoop102 subdir0]$ cat blk_1073741825
hadoop yarn
hadoop mapreduce
atguigu
atguigu
```

(3) 拼接

```
-rw-rw-r--. 1 atguigu atguigu 134217728 5 月 23 16:01 blk_1073741836
-rw-rw-r--. 1 atguigu atguigu 1048583 5 月 23 16:01 blk_1073741836_1012.meta
-rw-rw-r--. 1 atguigu atguigu 63439959 5 月 23 16:01 blk_1073741837
-rw-rw-r--. 1 atguigu atguigu 495635 5 月 23 16:01 blk_1073741837_1013.meta
[atguigu@hadoop102 subdir0]$ cat blk_1073741836>>tmp.tar.gz
[atguigu@hadoop102 subdir0]$ cat blk_1073741837>>tmp.tar.gz
[atguigu@hadoop102 subdir0]$ tar -zxvf tmp.tar.gz
```

(4) 下载

```
[atguigu@hadoop104 software]$ hadoop fs -get /jdk-8u212-linux-x64.tar.gz ./
```

(5) 执行 wordcount 程序

```
[atguigu@hadoop102 hadoop-3.1.3]$ hadoop jar
share/hadoop/mapreduce/hadoop-mapreduce-examples-3.1.3.jar
wordcount /input /output
```

## 4.2.6 集群启动/停止方式总结

### 1) 各个服务组件逐一启动/停止

(1) 分别启动/停止 HDFS 组件

```
hdfs --daemon start/stop namenode/datanode/secondarynamenode
```

(2) 启动/停止 YARN

```
yarn --daemon start/stop resourcemanager/nodemanager
```

### 2) 各个模块分开启动/停止 (配置 ssh 是前提) 常用

(1) 整体启动/停止 HDFS

```
start-dfs.sh/stop-dfs.sh
```

(2) 整体启动/停止 YARN

```
start-yarn.sh/stop-yarn.sh
```

## 4.2.7 配置历史服务器

为了查看程序的历史运行情况，需要配置一下历史服务器。具体配置步骤如下：

### 1) 配置 mapred-site.xml

```
[atguigu@hadoop102 hadoop]$ vim mapred-site.xml
```

在该文件里面增加如下配置。

```
<!-- 历史服务器端地址 -->
<property>
  <name>mapreduce.jobhistory.address</name>
  <value>hadoop102:10020</value>
</property>

<!-- 历史服务器 web 端地址 -->
<property>
  <name>mapreduce.jobhistory.webapp.address</name>
  <value>hadoop102:19888</value>
</property>
```

### 2) 分发配置



```
[atguigu@hadoop102 ~]$ xsync $HADOOP_HOME/etc/hadoop/mapred-site.xml
```

### 3) 在 hadoop102 启动历史服务器

```
[atguigu@hadoop102 ~]$ mapred --daemon start historyserver
```

### 4) 查看历史服务器是否启动

```
[atguigu@hadoop102 ~]$ jps
```

### 5) 查看 JobHistory

<http://hadoop102:19888/jobhistory>

## 4.2.8 配置日志的聚集

日志聚集概念：应用运行完成以后，将程序运行日志信息上传到 HDFS 系统上。

日志聚集功能好处：可以方便的查看到程序运行详情，方便开发调试。

**注意：开启日志聚集功能，需要重新启动 NodeManager 、 ResourceManager 和 HistoryServer。**

开启日志聚集功能具体步骤如下：

### 1) 配置 yarn-site.xml

```
[atguigu@hadoop102 ~]$ vim yarn-site.xml
```

在该文件里面增加如下配置。

```
<!-- 开启日志聚集功能 -->
<property>
  <name>yarn.log-aggregation-enable</name>
  <value>true</value>
</property>
<!-- 设置日志聚集服务器地址 -->
<property>
  <name>yarn.log.server.url</name>
  <value>http://hadoop102:19888/jobhistory/logs</value>
</property>
<!-- 设置日志保留时间为 7 天 -->
<property>
  <name>yarn.log-aggregation.retain-seconds</name>
  <value>604800</value>
</property>
```

### 2) 分发配置

```
[atguigu@hadoop102 ~]$ xsync $HADOOP_HOME/etc/hadoop/yarn-site.xml
```

### 3) 关闭 NodeManager 、 ResourceManager 和 HistoryServer

```
[atguigu@hadoop103 ~]$ stop-yarn.sh
[atguigu@hadoop102 ~]$ mapred --daemon stop historyserver
```

### 4) 启动 NodeManager 、 ResourceManager 和 HistoryServer

```
[atguigu@hadoop103 ~]$ start-yarn.sh
[atguigu@hadoop102 ~]$ mapred --daemon start historyserver
```

### 5) 删除 HDFS 上已经存在的输出文件


```
[atguigu@hadoop102 ~]$ hadoop fs -rm -r /output
```

## 6) 执行 WordCount 程序

```
[atguigu@hadoop102 ~]$ hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.1.3.jar wordcount /input /output
```

## 7) 查看日志,

<http://hadoop102:19888/jobhistory>



192.168.10.101:19888/jobhistory

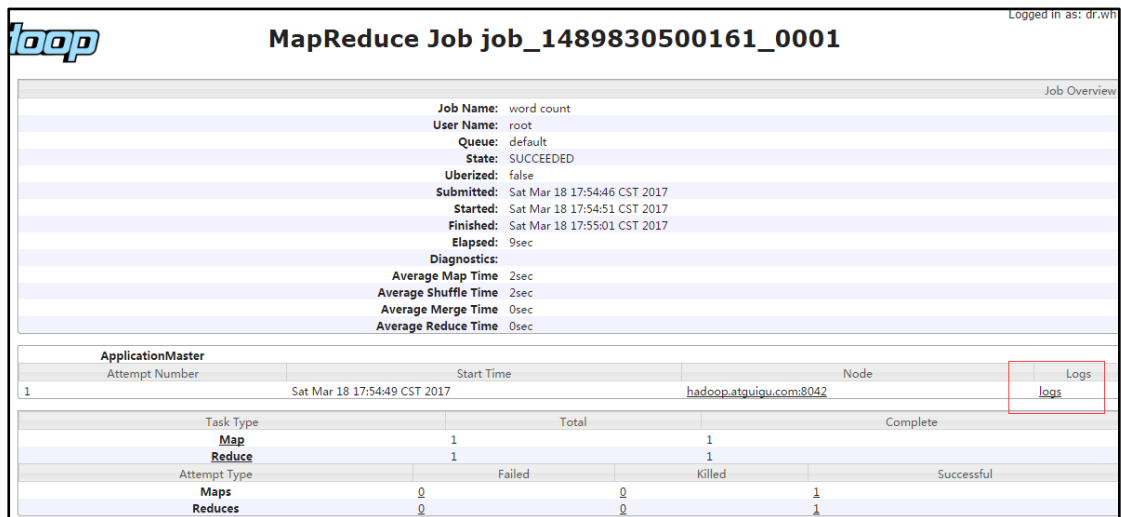
JobHistory

Retired Jobs

Show 20 entries

Submit Time	Start Time	Finish Time	Job ID	Name	User	Queue	State	Maps Total	Maps Completed	Reduces Total	Reduces Completed
2017.03.18 17:54:46 CST	2017.03.18 17:54:51 CST	2017.03.18 17:55:01 CST	job_1489830500161_0001	word count	root	default	SUCCEEDED	1	1	1	1
2017.03.18 17:20:31 CST	2017.03.18 17:20:39 CST	2017.03.18 17:20:50 CST	job_1489827711073_0001	word count	root	default	SUCCEEDED	1	1	1	1
2017.03.18 16:21:38 CST	2017.03.18 16:21:42 CST	2017.03.18 16:21:52 CST	job_1489820373751_0003	word count	root	default	SUCCEEDED	1	1	1	1
2017.03.18 15:20:57 CST	2017.03.18 15:30:02 CST	2017.03.18 15:30:12 CST	job_1489820373751_0002	word count	root	default	SUCCEEDED	1	1	1	1
2017.03.18 15:15:25 CST	2017.03.18 15:15:31 CST	2017.03.18 15:15:42 CST	job_1489820373751_0001	word count	root	default	SUCCEEDED	1	1	1	1

图 Job History



MapReduce Job job\_1489830500161\_0001

Job Overview

Job Name: word count

User Name: root

Queue: default

State: SUCCEEDED

Uberized: false

Submitted: Sat Mar 18 17:54:46 CST 2017

Started: Sat Mar 18 17:54:51 CST 2017

Finished: Sat Mar 18 17:55:01 CST 2017

Elapsed: 9sec

Diagnostics:

Average Map Time: 2sec

Average Shuffle Time: 2sec

Average Merge Time: 0sec

Average Reduce Time: 0sec

ApplicationMaster	Attempt Number	Start Time	Node	Logs
1		Sat Mar 18 17:54:49 CST 2017	hadoop.atguigu.com:8042	logs

Task Type	Total	Complete
Map	1	1
Reduce	1	1

Attempt Type	Failed	Killed	Successful
Maps	0	0	1
Reduces	0	0	1

图 job 运行情况



图 查看日志

## 4.2.9 编写 hadoop 集群常用脚本

### 1) 查看三台服务器 java 进程脚本: jpsall

```
[atguigu@hadoop102 ~]$ cd /home/atguigu/bin
[atguigu@hadoop102 ~]$ vim jpsall
```

然后输入

```
#!/bin/bash
for host in hadoop102 hadoop103 hadoop104
do
    echo ===== $host =====
    ssh $host jps $@ | grep -v Jps
done
```

保存后退出, 然后赋予脚本执行权限

```
[atguigu@hadoop102 bin]$ chmod +x jpsall
```

### 2) hadoop 集群启停脚本 (包含 hdfs, yarn, historyserver): myhadoop.sh

```
[atguigu@hadoop102 ~]$ cd /home/atguigu/bin
[atguigu@hadoop102 ~]$ vim myhadoop.sh
```

然后输入

```
#!/bin/bash
if [ $# -lt 1 ]
then
    echo "No Args Input..."
    exit ;
fi
case $1 in
"start")
    echo " ===== 启动 hadoop 集群 ====="

    echo " ----- 启动 hdfs -----"
    ssh hadoop102 "/opt/module/hadoop-3.1.3/sbin/start-dfs.sh"
    echo " ----- 启动 yarn -----"
    ssh hadoop103 "/opt/module/hadoop-3.1.3/sbin/start-yarn.sh"
    echo " ----- 启动 historyserver -----"
    ssh hadoop102 "/opt/module/hadoop-3.1.3/bin/mapred --daemon start
historyserver"
;;
```

```
"stop")
    echo " ===== 关闭 hadoop 集群 ====="

    echo " ----- 关闭 historyserver -----"
    ssh hadoop102 "/opt/module/hadoop-3.1.3/bin/mapred --daemon stop
historyserver"
    echo " ----- 关闭 yarn -----"
    ssh hadoop103 "/opt/module/hadoop-3.1.3/sbin/stop-yarn.sh"
    echo " ----- 关闭 hdfs -----"
    ssh hadoop102 "/opt/module/hadoop-3.1.3/sbin/stop-dfs.sh"

;;
*)
    echo "Input Args Error..."
;;
esac
```

保存后退出，然后赋予脚本执行权限

```
[atguigu@hadoop102 bin]$ chmod +x myhadoop.sh
```

### 3) 分发/home/atguigu/bin 目录，保证自定义脚本在三台机器上都可以使用

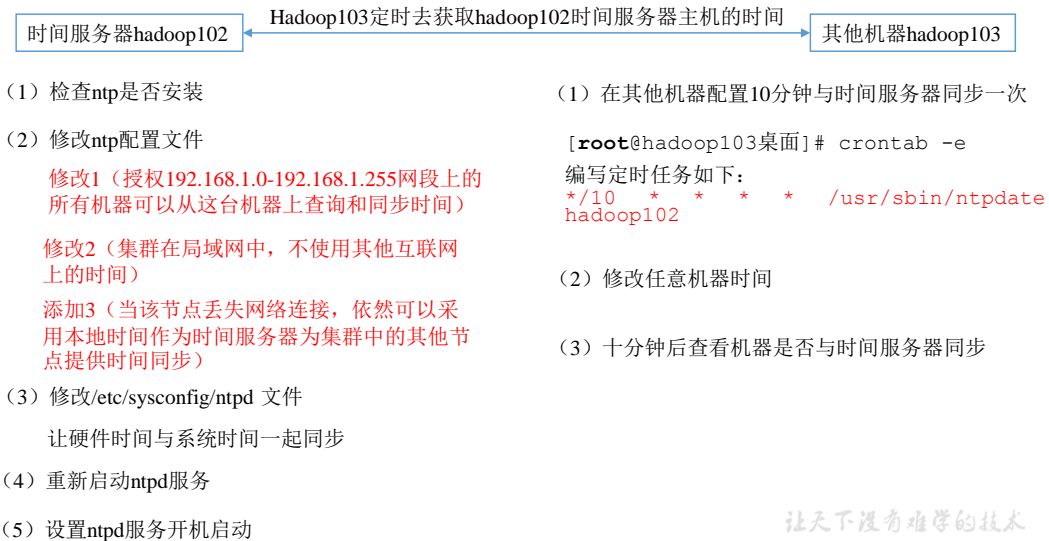
```
[atguigu@hadoop102 ~]$ xsync /home/atguigu/bin/
```

## 4.2.10 集群时间同步

时间同步的方式：找一个机器，作为时间服务器，所有的机器与这台集群时间进行定时的同步，比如，每隔十分钟，同步一次时间。



### 集群时间同步



### 配置时间同步具体实操：

#### 1) 时间服务器配置（必须 root 用户）

(0) 查看所有节点 ntpd 服务状态和开机自启动状态

```
[atguigu@hadoop102 ~]$ sudo systemctl status ntpd
[atguigu@hadoop102 ~]$ sudo systemctl is-enabled ntpd
```

(1) 在所有节点关闭 ntp 服务和自启动

```
[atguigu@hadoop102 ~]$ sudo systemctl stop ntpd
[atguigu@hadoop102 ~]$ sudo systemctl disable ntpd
```

(2) 修改 hadoop102 的 ntp.conf 配置文件

```
[atguigu@hadoop102 ~]$ sudo vim /etc/ntp.conf
```

修改内容如下

a) 修改 1 (授权 192.168.1.0-192.168.1.255 网段上的所有机器可以从这台机器上查询和同步时间)

```
#restrict 192.168.1.0 mask 255.255.255.0 nomodify notrap
```

为 restrict 192.168.1.0 mask 255.255.255.0 nomodify notrap

b) 修改 2 (集群在局域网中, 不使用其他互联网上的时间)

```
server 0.centos.pool.ntp.org iburst
server 1.centos.pool.ntp.org iburst
server 2.centos.pool.ntp.org iburst
server 3.centos.pool.ntp.org iburst
```

为

```
#server 0.centos.pool.ntp.org iburst
#server 1.centos.pool.ntp.org iburst
#server 2.centos.pool.ntp.org iburst
#server 3.centos.pool.ntp.org iburst
```

c) 添加 3 (当该节点丢失网络连接, 依然可以采用本地时间作为时间服务器为集群中的其他节点提供时间同步)

```
server 127.127.1.0
fudge 127.127.1.0 stratum 10
```

(3) 修改 hadoop102 的/etc/sysconfig/ntpd 文件

```
[atguigu@hadoop102 ~]$ sudo vim /etc/sysconfig/ntpd
```

增加内容如下 (让硬件时间与系统时间一起同步)

```
SYNC_HWCLOCK=yes
```

(4) 重新启动 ntpd 服务

```
[atguigu@hadoop102 ~]$ sudo systemctl start ntpd
```

(5) 设置 ntpd 服务开机启动

```
[atguigu@hadoop102 ~]$ sudo systemctl enable ntpd
```

## 2) 其他机器配置 (必须 root 用户)

(1) 在其他机器配置 10 分钟与时间服务器同步一次

```
[atguigu@hadoop103 ~]$ sudo crontab -e
```

编写定时任务如下:

```
*/10 * * * * /usr/sbin/ntpdate hadoop102
```

(2) 修改任意机器时间

```
[atguigu@hadoop103 ~]$ sudo date -s "2017-9-11 11:11:11"
```

(3) 十分钟后查看机器是否与时间服务器同步

```
[atguigu@hadoop103 ~]$ sudo date
```

说明：测试的时候可以将 10 分钟调整为 1 分钟，节省时间。

## 第 5 章 Hadoop 编译源码【了解】

### 5.1 前期准备工作

#### 1) CentOS 联网

配置 CentOS 能连接外网。Linux 虚拟机 ping [www.baidu.com](http://www.baidu.com) 是畅通的

注意：采用 root 角色编译，减少文件夹权限出现问题

#### 2) jar 包准备(hadoop 源码、JDK8、maven、ant 、protobuf)

- (1) hadoop-3.1.3-src.tar.gz
- (2) jdk-8u212-linux-x64.tar.gz
- (3) apache-maven-3.6.3-bin.tar.gz
- (4) protobuf-2.5.0.tar.gz (序列化的框架)
- (5) cmake-3.13.1.tar.gz

### 5.2 Jar 包安装

注意：所有操作必须在 root 用户下完成

#### 1) 上传软件包到指定的目录 ,例如 /opt/software/hadoop\_source

```
[root@hadoop101 hadoop_source]$ pwd
/opt/software/hadoop_source
[root@hadoop101 hadoop_source]$ ll
总用量 55868
-rw-rw-r--. 1 atguigu atguigu 9506321 3月 28 13:23 apache-maven-3.6.3-
bin.tar.gz
-rw-rw-r--. 1 atguigu atguigu 8614663 3月 28 13:23 cmake-3.13.1.tar.gz
-rw-rw-r--. 1 atguigu atguigu 29800905 3月 28 13:23 hadoop-3.1.3-
src.tar.gz
-rw-rw-r--. 1 atguigu atguigu 2401901 3月 28 13:23 protobuf-
2.5.0.tar.gz
```

#### 2) 解压软件包指定的目录，例如： /opt/module/Hadoop\_source

```
[atguigu@hadoop101 hadoop_source]$ tar -zxvf apache-maven-3.6.3-
bin.tar.gz -C /opt/module/hadoop_source/

[atguigu@hadoop101 hadoop_source]$ tar -zxvf cmake-3.13.1.tar.gz -C
/opt/module/hadoop_source/

[atguigu@hadoop101 hadoop_source]$ tar -zxvf hadoop-3.1.3-src.tar.gz -C
/opt/module/hadoop_source/

[atguigu@hadoop101 hadoop_source]$ tar -zxvf protobuf-2.5.0.tar.gz -C
/opt/module/hadoop_source/

[atguigu@hadoop101 hadoop_source]$ pwd
/opt/module/hadoop_source

[atguigu@hadoop101 hadoop_source]$ ll
总用量 20
drwxrwxr-x. 6 atguigu atguigu 4096 3月 28 13:25 apache-maven-3.6.3
drwxr-xr-x. 15 root root 4096 3月 28 13:43 cmake-3.13.1
drwxr-xr-x. 18 atguigu atguigu 4096 9月 12 2019 hadoop-3.1.3-src
```



```
drwxr-xr-x. 10 atguigu atguigu 4096 3月 28 13:44 protobuf-2.5.0
```

3) 确认 Java 已安装且配置好环境变量, 安装完后验证

```
[atguigu@hadoop101 hadoop_source]$ java -version
java version "1.8.0_212"
Java(TM) SE Runtime Environment (build 1.8.0_212-b10)
Java HotSpot(TM) 64-Bit Server VM (build 25.212-b10, mixed mode)
```

4) 配置 maven 环境变量, maven 镜像, 并验证

```
#配置 maven 的环境变量
[root@hadoop101 hadoop_source]# vim /etc/profile
#MAVEN_HOME
MAVEN_HOME=/opt/module/hadoop_source/apache-maven-3.6.3
PATH=$PATH:$JAVA_HOME/bin:$MAVEN_HOME/bin

[root@hadoop101 hadoop_source]# source /etc/profile

#修改 maven 的镜像
[root@hadoop101 apache-maven-3.6.3]# vi conf/settings.xml

# 在 mirrors 节点中添加阿里云镜像

<mirrors>
    <mirror>
        <id>nexus-aliyun</id>
        <mirrorOf>central</mirrorOf>
        <name>Nexus aliyun</name>

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

[root@hadoop101 hadoop_source]# mvn -version
Apache Maven 3.6.3 (cecedd343002696d0abb50b32b541b8a6ba2883f)
Maven home: /opt/module/hadoop_source/apache-maven-3.6.3
Java version: 1.8.0_212, vendor: Oracle Corporation, runtime:
/opt/module/jdk1.8.0_212/jre
Default locale: zh_CN, platform encoding: UTF-8
OS name: "linux", version: "3.10.0-862.el7.x86_64", arch: "amd64",
family: "unix"
```

5) 安装相关的依赖(注意安装顺序不可乱,可能会出现依赖找不到问题)

```
# 安装 gcc make
[root@hadoop101 hadoop_source]# yum install -y gcc* make

#安装压缩工具
[root@hadoop101 hadoop_source]# yum -y install snappy* bzip2* lzo* zlib*
lz4* gzip*

#安装一些基本工具
[root@hadoop101 hadoop_source]# yum -y install openssl* svn ncurses*
autoconf automake libtool

#安装扩展源,才可安装 zstd
[root@hadoop101 hadoop_source]# yum -y install epel-release
#安装 zstd
[root@hadoop101 hadoop_source]# yum -y install *zstd*
```

6) 手动安装 cmake

1. 在解压好的 cmake 目录下,执行 ./bootstrap 进行编译,此过程需一小会时间耐心

等待.

```
[atguigu@hadoop101 cmake-3.13.1]$ pwd
/opt/module/hadoop_source/cmake-3.13.1
[atguigu@hadoop101 cmake-3.13.1]$ ./bootstrap
```

## 2. 执行安装

```
[atguigu@hadoop101 cmake-3.13.1]$ make && make install
```

## 3. 验证安装是否成功

```
[atguigu@hadoop101 cmake-3.13.1]$ cmake -version
cmake version 3.13.1
```

```
CMake suite maintained and supported by Kitware (kitware.com/cmake).
```

## 7) 安装 protobuf，进入到解压后的 protobuf 目录

```
[atguigu@hadoop101 protobuf-2.5.0]$ pwd
/opt/module/hadoop_source/protobuf-2.5.0

#依次执行下列命令 --prefix 指定安装到当前目录
[atguigu@hadoop101 protobuf-2.5.0]$ ./configure --
prefix=/opt/module/hadoop_source/protobuf-2.5.0
[atguigu@hadoop101 protobuf-2.5.0]$ make && make install

#配置环境变量
[atguigu@hadoop101 protobuf-2.5.0]$ vim /etc/profile

PROTOC_HOME=/opt/module/hadoop_source/protobuf-2.5.0
PATH=$PATH:$JAVA_HOME/bin:$MAVEN_HOME/bin:$PROTOC_HOME/bin

#验证
[atguigu@hadoop101 protobuf-2.5.0]$ source /etc/profile
[atguigu@hadoop101 protobuf-2.5.0]$ protoc --version
libprotoc 2.5.0
```

## 8) 到此，软件包安装配置工作完成。

# 5.3编译源码

## 1) 进入解压后的 hadoop 源码目录下

```
[atguigu@hadoop101 hadoop-3.1.3-src]$ pwd
/opt/module/hadoop_source/hadoop-3.1.3-src

#开始编译
[atguigu@hadoop101 hadoop-3.1.3-src]$ mvn clean package -DskipTests -
Pdist,native -Dtar
```

等等等.....等待，第一次编译需要下载很多依赖 jar 包，编译时间会很久，预计 1 小时左右,最终成功是全部 SUCCESS，爽!!! 如图 1-1

**[INFO] Reactor Summary for Apache Hadoop Main 3.1.3:****[INFO]**

<b>[INFO]</b> Apache Hadoop Main .....	<b>SUCCESS</b>	[ 1.591 s]
<b>[INFO]</b> Apache Hadoop Build Tools .....	<b>SUCCESS</b>	[ 2.804 s]
<b>[INFO]</b> Apache Hadoop Project POM .....	<b>SUCCESS</b>	[ 1.329 s]
<b>[INFO]</b> Apache Hadoop Annotations .....	<b>SUCCESS</b>	[ 2.903 s]
<b>[INFO]</b> Apache Hadoop Assemblies .....	<b>SUCCESS</b>	[ 0.136 s]
<b>[INFO]</b> Apache Hadoop Project Dist POM .....	<b>SUCCESS</b>	[ 1.485 s]
<b>[INFO]</b> Apache Hadoop Maven Plugins .....	<b>SUCCESS</b>	[ 4.168 s]
<b>[INFO]</b> Apache Hadoop MinikDC .....	<b>SUCCESS</b>	[ 1.952 s]
<b>[INFO]</b> Apache Hadoop Auth .....	<b>SUCCESS</b>	[ 5.617 s]
<b>[INFO]</b> Apache Hadoop Auth Examples .....	<b>SUCCESS</b>	[ 2.594 s]
<b>[INFO]</b> Apache Hadoop Common .....	<b>SUCCESS</b>	[ 59.521 s]
<b>[INFO]</b> Apache Hadoop NFS .....	<b>SUCCESS</b>	[ 4.810 s]
<b>[INFO]</b> Apache Hadoop KMS .....	<b>SUCCESS</b>	[ 4.260 s]
<b>[INFO]</b> Apache Hadoop Common Project .....	<b>SUCCESS</b>	[ 0.041 s]
<b>[INFO]</b> Apache Hadoop HDFS Client .....	<b>SUCCESS</b>	[ 27.089 s]
<b>[INFO]</b> Apache Hadoop HDFS .....	<b>SUCCESS</b>	[ 53.519 s]
<b>[INFO]</b> Apache Hadoop HDFS Native Client .....	<b>SUCCESS</b>	[ 4.357 s]
<b>[INFO]</b> Apache Hadoop HttpFS .....	<b>SUCCESS</b>	[ 6.029 s]
<b>[INFO]</b> Apache Hadoop HDFS-NFS .....	<b>SUCCESS</b>	[ 3.462 s]
<b>[INFO]</b> Apache Hadoop HDFS-RBF .....	<b>SUCCESS</b>	[ 17.960 s]
<b>[INFO]</b> Apache Hadoop HDFS Project .....	<b>SUCCESS</b>	[ 0.055 s]
<b>[INFO]</b> Apache Hadoop YARN .....	<b>SUCCESS</b>	[ 0.034 s]
<b>[INFO]</b> Apache Hadoop YARN API .....	<b>SUCCESS</b>	[ 15.652 s]
<b>[INFO]</b> Apache Hadoop YARN Common .....	<b>SUCCESS</b>	[ 33.167 s]
<b>[INFO]</b> Apache Hadoop YARN Registry .....	<b>SUCCESS</b>	[ 5.440 s]
<b>[INFO]</b> Apache Hadoop YARN Server .....	<b>SUCCESS</b>	[ 0.066 s]
<b>[INFO]</b> Apache Hadoop YARN Server Common .....	<b>SUCCESS</b>	[ 10.413 s]
<b>[INFO]</b> Apache Hadoop YARN NodeManager .....	<b>SUCCESS</b>	[ 30.261 s]
<b>[INFO]</b> Apache Hadoop YARN Web Proxy .....	<b>SUCCESS</b>	[ 3.776 s]
<b>[INFO]</b> Apache Hadoop YARN ApplicationHistoryService .....	<b>SUCCESS</b>	[ 5.905 s]
<b>[INFO]</b> Apache Hadoop YARN Timeline Service .....	<b>SUCCESS</b>	[ 5.206 s]
<b>[INFO]</b> Apache Hadoop YARN ResourceManager .....	<b>SUCCESS</b>	[ 22.169 s]
<b>[INFO]</b> Apache Hadoop YARN Server Tests .....	<b>SUCCESS</b>	[ 1.235 s]
<b>[INFO]</b> Apache Hadoop YARN Client .....	<b>SUCCESS</b>	[ 5.872 s]
<b>[INFO]</b> Apache Hadoop YARN SharedCacheManager .....	<b>SUCCESS</b>	[ 2.887 s]
<b>[INFO]</b> Apache Hadoop YARN Timeline Plugin Storage .....	<b>SUCCESS</b>	[ 3.384 s]
<b>[INFO]</b> Apache Hadoop YARN TimelineService HBase Backend ...	<b>SUCCESS</b>	[ 0.030 s]
<b>[INFO]</b> Apache Hadoop YARN TimelineService HBase Common ...	<b>SUCCESS</b>	[ 5.268 s]
<b>[INFO]</b> Apache Hadoop YARN TimelineService HBase Client ...	<b>SUCCESS</b>	[ 4.341 s]
<b>[INFO]</b> Apache Hadoop YARN TimelineService HBase Servers ...	<b>SUCCESS</b>	[ 0.041 s]
<b>[INFO]</b> Apache Hadoop YARN TimelineService HBase Server 1.2	<b>SUCCESS</b>	[ 5.277 s]
<b>[INFO]</b> Apache Hadoop YARN TimelineService HBase tests .....	<b>SUCCESS</b>	[ 2.031 s]
<b>[INFO]</b> Apache Hadoop YARN Router .....	<b>SUCCESS</b>	[ 5.126 s]

[INFO]	Apache Hadoop YARN Applications .....	SUCCESS	[ 0.031 s]
[INFO]	Apache Hadoop YARN DistributedShell .....	SUCCESS	[ 3.033 s]
[INFO]	Apache Hadoop YARN Unmanaged Am Launcher .....	SUCCESS	[ 2.014 s]
[INFO]	Apache Hadoop MapReduce Client .....	SUCCESS	[ 0.179 s]
[INFO]	Apache Hadoop MapReduce Core .....	SUCCESS	[ 17.479 s]
[INFO]	Apache Hadoop MapReduce Common .....	SUCCESS	[ 13.605 s]
[INFO]	Apache Hadoop MapReduce Shuffle .....	SUCCESS	[ 4.207 s]
[INFO]	Apache Hadoop MapReduce App .....	SUCCESS	[ 8.606 s]
[INFO]	Apache Hadoop MapReduce HistoryServer .....	SUCCESS	[ 5.052 s]
[INFO]	Apache Hadoop MapReduce JobClient .....	SUCCESS	[ 5.550 s]
[INFO]	Apache Hadoop Mini-Cluster .....	SUCCESS	[ 0.829 s]
[INFO]	Apache Hadoop YARN Services .....	SUCCESS	[ 0.025 s]
[INFO]	Apache Hadoop YARN Services Core .....	SUCCESS	[ 2.852 s]
[INFO]	Apache Hadoop YARN Services API .....	SUCCESS	[ 1.556 s]
[INFO]	Apache Hadoop YARN Site .....	SUCCESS	[ 0.037 s]
[INFO]	Apache Hadoop YARN UI .....	SUCCESS	[ 0.064 s]
[INFO]	Apache Hadoop YARN Project .....	SUCCESS	[ 8.548 s]
[INFO]	Apache Hadoop MapReduce HistoryServer Plugins .....	SUCCESS	[ 1.850 s]
[INFO]	Apache Hadoop MapReduce NativeTask .....	SUCCESS	[ 24.779 s]
[INFO]	Apache Hadoop MapReduce Uploader .....	SUCCESS	[ 1.955 s]
[INFO]	Apache Hadoop MapReduce Examples .....	SUCCESS	[ 4.581 s]
[INFO]	Apache Hadoop MapReduce .....	SUCCESS	[ 4.427 s]
[INFO]	Apache Hadoop MapReduce Streaming .....	SUCCESS	[ 4.486 s]
[INFO]	Apache Hadoop Distributed Copy .....	SUCCESS	[ 4.940 s]
[INFO]	Apache Hadoop Archives .....	SUCCESS	[ 2.469 s]
[INFO]	Apache Hadoop Archive Logs .....	SUCCESS	[ 2.046 s]
[INFO]	Apache Hadoop Rumen .....	SUCCESS	[ 5.017 s]
[INFO]	Apache Hadoop Gridmix .....	SUCCESS	[ 3.819 s]
[INFO]	Apache Hadoop Data Join .....	SUCCESS	[ 2.088 s]
[INFO]	Apache Hadoop Extras .....	SUCCESS	[ 2.221 s]
[INFO]	Apache Hadoop Pipes .....	SUCCESS	[ 4.266 s]
[INFO]	Apache Hadoop OpenStack support .....	SUCCESS	[ 3.937 s]
[INFO]	Apache Hadoop Amazon Web Services support .....	SUCCESS	[47:44 min]
[INFO]	Apache Hadoop Kafka Library support .....	SUCCESS	[ 7.801 s]
[INFO]	Apache Hadoop Azure support .....	SUCCESS	[ 13.740 s]
[INFO]	Apache Hadoop Aliyun OSS support .....	SUCCESS	[ 34.460 s]
[INFO]	Apache Hadoop Client Aggregator .....	SUCCESS	[ 1.460 s]
[INFO]	Apache Hadoop Scheduler Load Simulator .....	SUCCESS	[ 4.471 s]
[INFO]	Apache Hadoop Resource Estimator Service .....	SUCCESS	[ 10.356 s]
[INFO]	Apache Hadoop Azure Data Lake support .....	SUCCESS	[ 13.289 s]
[INFO]	Apache Hadoop Image Generation Tool .....	SUCCESS	[ 3.430 s]
[INFO]	Apache Hadoop Tools Dist .....	SUCCESS	[ 7.640 s]
[INFO]	Apache Hadoop Tools .....	SUCCESS	[ 0.025 s]
[INFO]	Apache Hadoop Client API .....	SUCCESS	[02:02 min]
[INFO]	Apache Hadoop Client Runtime .....	SUCCESS	[01:33 min]
[INFO]	Apache Hadoop Client Packaging Invariants .....	SUCCESS	[ 1.507 s]
[INFO]	Apache Hadoop Client Test Minicluster .....	SUCCESS	[02:52 min]
[INFO]	Apache Hadoop Client Packaging Invariants for Test .....	SUCCESS	[ 0.115 s]

```
[INFO] Apache Hadoop Client Packaging Integration Tests ... SUCCESS [ 0.080 s]
[INFO] Apache Hadoop Distribution ..... SUCCESS [ 27.180 s]
[INFO] Apache Hadoop Client Modules ..... SUCCESS [ 0.044 s]
[INFO] Apache Hadoop Cloud Storage ..... SUCCESS [ 0.886 s]
[INFO] Apache Hadoop Cloud Storage Project ..... SUCCESS [ 0.023 s]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 01:05 h
[INFO] Finished at: 2020-03-29T16:24:41+08:00
[INFO] -----
```

图 1-1 编译成功

2) 成功的 64 位 hadoop 包在/opt/hadoop-3.1.3-src/hadoop-dist/target 下

```
[root@hadoop101 target]# pwd
/opt/hadoop-3.1.3-src/hadoop-dist/target

[root@hadoop200 target]# pwd
/opt/module/hadoop_source/hadoop-3.1.3-src/hadoop-dist/target
[root@hadoop200 target]# ll
总用量 288060
drwxr-xr-x. 2 root root      4096 3月  29 16:24 antrun
drwxr-xr-x. 3 root root      4096 3月  29 16:24 classes
drwxr-xr-x. 9 root root      4096 3月  29 16:24 hadoop-3.1.3
-rw-r--r--. 1 root root 294945821 3月  29 16:24 hadoop-3.1.3.tar.gz
drwxr-xr-x. 3 root root      4096 3月  29 16:24 maven-shared-archive-resources
drwxr-xr-x. 3 root root      4096 3月  29 16:24 test-classes
drwxr-xr-x. 2 root root      4096 3月  29 16:24 test-dir
```

## 第 6 章 常见错误及解决方案

1) 防火墙没关闭、或者没有启动 YARN

```
INFO client.RMProxy: Connecting to ResourceManager at hadoop108/192.168.10.108:8032
```

2) 主机名称配置错误

3) IP 地址配置错误

4) ssh 没有配置好

5) root 用户和 atguigu 两个用户启动集群不统一

6) 配置文件修改不细心

7) 未编译源码

```
Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
```

```
17/05/22 15:38:58 INFO client.RMProxy: Connecting to ResourceManager at
hadoop108/192.168.10.108:8032
```

## 8) 不识别主机名称

```
java.net.UnknownHostException: hadoop102: hadoop102
    at
    java.net.InetAddress.getLocalHost(InetAddress.java:1475)
    at
    org.apache.hadoop.mapreduce.JobSubmitter.submitJobInternal(Job
    Submitter.java:146)
    at org.apache.hadoop.mapreduce.Job$10.run(Job.java:1290)
    at org.apache.hadoop.mapreduce.Job$10.run(Job.java:1287)
    at java.security.AccessController.doPrivileged(Native
    Method)
    at javax.security.auth.Subject.doAs(Subject.java:415)
```

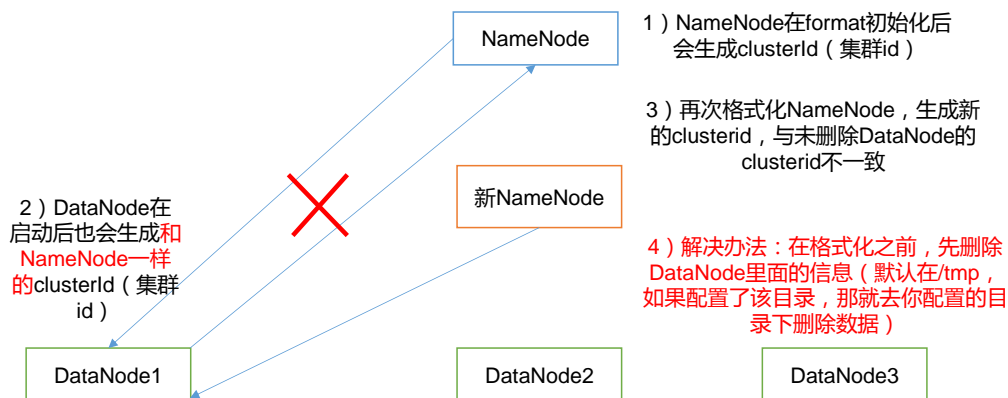
解决办法:

- (1) 在/etc/hosts 文件中添加 192.168.1.102 hadoop102
- (2) 主机名称不要起 hadoop hadoop000 等特殊名称

## 9) DataNode 和 NameNode 进程同时只能工作一个。



## DataNode和NameNode进程同时只能有一个工作问题分析



让天下没有难学的技术

## 10) 执行命令不生效, 粘贴 word 中命令时, 遇到-和长-没区分开。导致命令失效

解决办法: 尽量不要粘贴 word 中代码。

11) jps 发现进程已经没有, 但是重新启动集群, 提示进程已经开启。原因是在 linux 的根目录下/tmp 目录中存在启动的进程临时文件, 将集群相关进程删除掉, 再重新启动集群。

## 12) jps 不生效。

原因: 全局变量 hadoop java 没有生效。解决办法: 需要 source /etc/profile 文件。

## 13) 8088 端口连接不上

```
[atguigu@hadoop102 桌面]$ cat /etc/hosts
```

注释掉如下代码

更多 [Java](#) - [大数据](#) - [前端](#) - [python](#) 人工智能资料下载, 可百度访问: [尚硅谷官网](#)



```
#127.0.0.1    localhost localhost.localdomain localhost4 localhost4.localdomain4
```

```
#::1         hadoop102
```