## 分布式技术

### 为什么需要分布式

**计算问题**

无论是我们在学校刚开始学编程，还是在刚参加工作开始处理实际问题，写出来的程序都是很简单的。因为面对的问题很简单。以处理数据为例，可能只是把一个几十K的文件解析下，然后生成一个词频分析的报告。很简单的程序，十几行甚至几行就搞定了。

直到有一天，给你扔过来1000个文件，有些还特别大，好几百M了。你用之前的程序一跑，发现跑的时间有点长。于是想要去优化下。1000 个文件，互相还没业务联系，用多线程呀，一个线程处理一个文件，结果再汇总就搞定了。如果多线程效果不够好，比如像 Python 的多线程，没法利用多核的威力，那就用多进程。

无论是线程、进程，本质上，目的都是为了计算的并行化，解决的是算的慢的问题。而如果计算量足够大，就算榨干了机器的计算能力，也算不过来，咋办？

一台机器不够，那就多搞几台机器嘛。所以就从多线程/进程的计算并行化，进化到计算的分布式化（当然，分布式一定程度上也是并行化）。

**存储问题**

另一方面，如果处理的数据有10T，而你手上的机器只有500G 的硬盘，怎么办？

一种办法是纵向扩展，搞一台几十T硬盘的机器；另一种是横向扩展，多搞几台机器，分散着放。前者很容易到瓶颈，毕竟数据无限，而一台机器的容量有限，所以在大数据量的情况下，只能选后者。把数据分散到多台机器，本质上解决的是存不下的问题。

同时，刚才提到计算分布式化后，总不能所以程序都去同一台机器读数据吧，这样效率必然会受到单台机器性能的拖累，比如磁盘 IO、网络带宽等，也就逼着数据存储也要分散到各个机器去了。基于这两个原因，数据存储也分布式起来了。

### 分布式系统概述

**分布式系统是一个硬件或软件组件分布在不同的网络计算机上，彼此之间仅仅通过消息传递进行通信和协调的系统。**简单来说就是一群独立计算机集合共同对外提供服务，但是对于系统的用户来说，就像是一台计算机在提供服务一样。

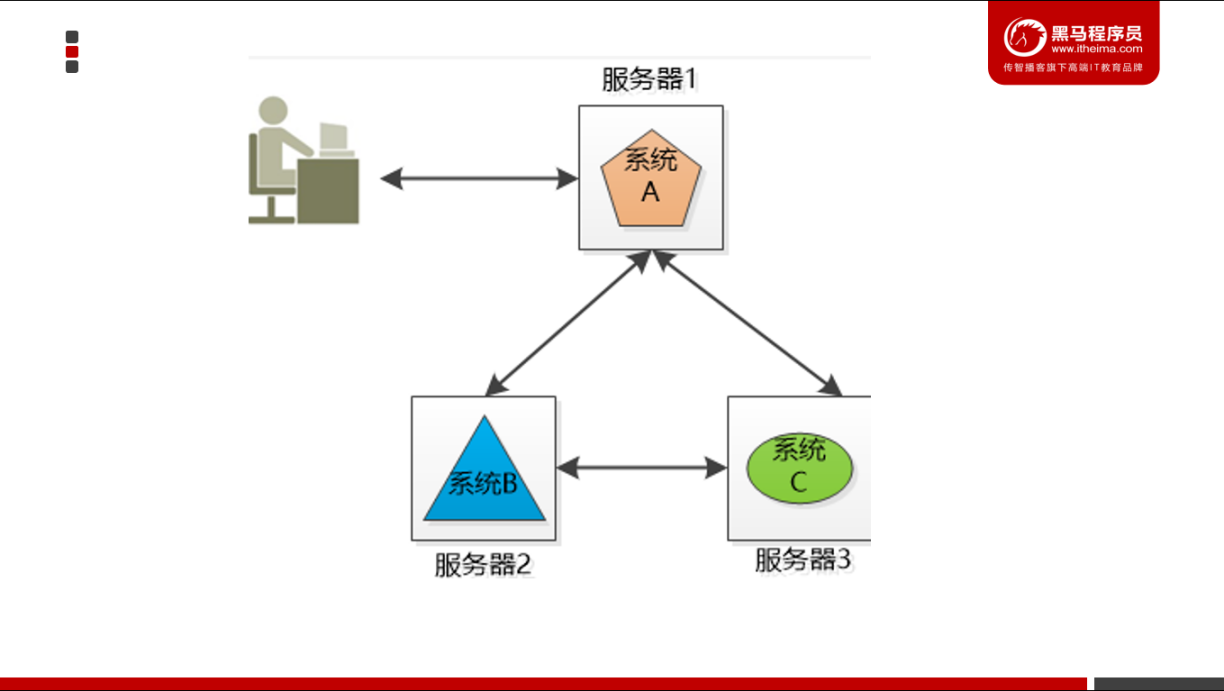
分布式意味着可以采用更多的普通计算机（相对于昂贵的大型机）组成分布式集群对外提供服务。计算机越多，CPU、内存、存储资源等也就越多，能够处理的并发访问量也就越大。

从分布式系统的概念中我们知道，各个主机之间通信和协调主要通过网络进行，所以，分布式系统中的计算机在空间上几乎没有任何限制，这些计算机可能被放在不同的机柜上，也可能被部署在不同的机房中，还可能在不同的城市中，对于大型的网站甚至可能分布在不同的国家和地区。

### 分布式实现方案

**分布式系统**

小明的公司又3个系统：系统A，系统B和系统C，这三个系统所做的业务不同，被部署在3个独立的机器上运行，他们之间互相调用（当然是跨域网络的），通力合作完成公司的业务流程。



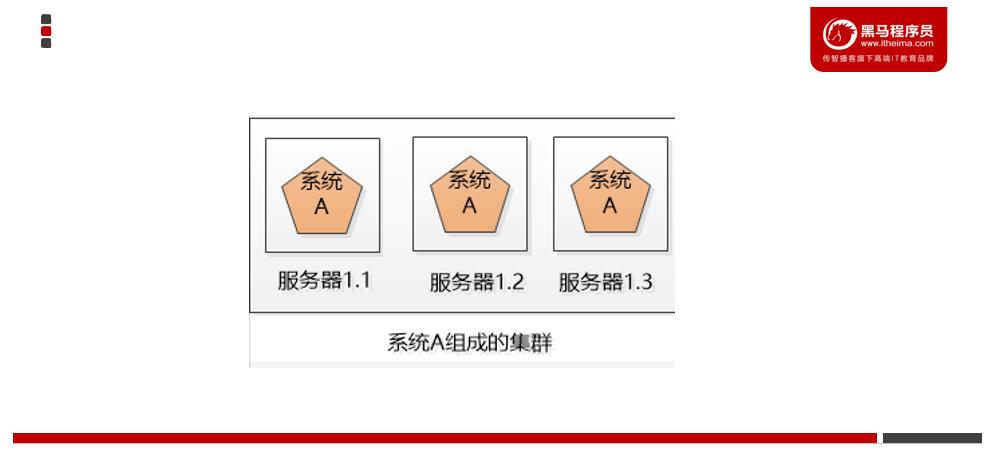
将不同的业务分部在不同的地方，就构成了一个分布式的系统，现在问题来了，系统A是整个分布式系统的脸面，用户直接访问，用户访问量大的时候要么是速度巨慢，要么直接挂掉，怎么办？

由于系统A只有一份，所以会引起单点失败。

**集群（Cluster）**

小明的公司不差钱，就多买几台机器吧， 小明把系统A一下子部署了好几份（例如下图的3个服务器），每一份都是系统A的一个实例，对外提供同样的服务，这样，就不怕其中一个坏掉了，还有另外两个呢。

这三个服务器的系统就组成了一个集群。



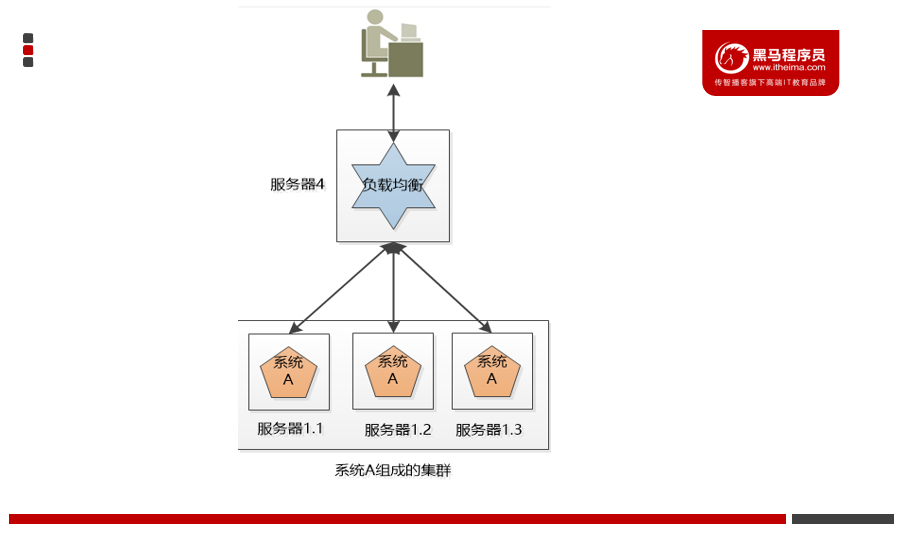
可是对用户来说，一下子出现这么多系统A，每个系统的IP地址都不一样，到底访问哪一个呢？

如果所有人都访问服务器1.1，那服务器1.1会被累死，剩下两个闲死，成了浪费钱的摆设。

**负载均衡（Load Balancer）**

小明要尽可能的让3个机器上的系统A工作均衡一些，比如有3万个请求，那就让3个服务器各处理1万个（理想情况），这叫负载均衡

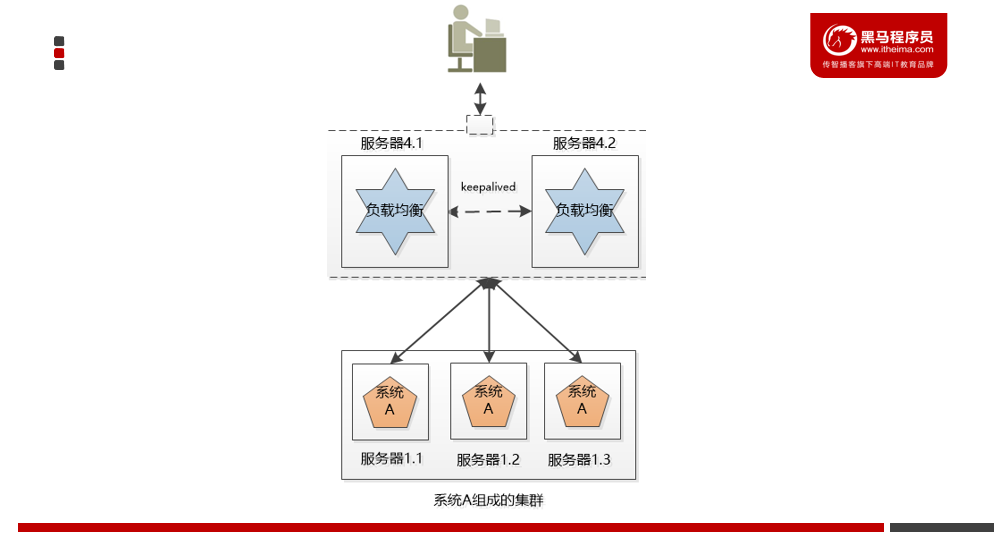
很明显，这个负载均衡的工作最好独立出来，放到独立的服务器上（例如nginx）：



后来小明发现，这个负载均衡的服务器虽然工作内容简单，就是拿到请求，分发请求，但是它还是有可能挂掉，单点失败还是会出现。没办法，只好把负载均衡也搞成一个集群，这个集群和系统A的集群有两点不同：

1.我们可以用某种办法，让这个机器对外只提供一个IP地址，也就是用户看到的好像只有一个机器。

2.同一时刻，我们只让一个负载均衡的机器工作，另外一个原地待命，如果工作的那个拐到了，待命的那个就顶上去。



**弹性(伸缩性)**

如果3个系统A的实例还是满足不了大量请求，例如双十一，可以申请增加服务器，双十一过后，新增的服务器闲置，成了摆设，于是小明决定尝试云计算，在云端可以轻松的创建，删除虚拟的服务器，那样就可以轻松的随着用户的请求动图的增减服务器了。

**失效转移**

上面的系统看起来很美好，但是做了一个不切实际的假设：所有的服务都是无状态的，换句话说，假设用户的两次请求直接是没有关联的。但是现实是，大部分服务都是有状态的，例如购物车。

用户访问系统，在服务器上创建了一个购物车，并向其中加了几个商品，然后服务器1.1挂掉了，用户后续访问就找不到服务器1.1了，这时候就要做失效转移，让另外几个服务器去接管，去处理用户的请求。

可是问题来了，在服务器1.2,1.3上有用户的购物车吗？如果没有，用户就会抱怨，我刚创建的购物车哪里去了？还有更严重的，假设用户登录过得信息保存到了该服务器1.1上登录的，用户登录过的信息保存到了该服务器的session中，现在这个服务器挂了，用的session就不见了，会把用户踢到了登录界面，让用户再次登录！

处理不好状态的问题，集群的威力就大打折扣，无法完成真正的失效转移，甚至无法使用。

怎么办？

一种办法是把状态信息在集群的各个服务器之间复制，让集群的各个服务器达成一致。

还有一种办法， 就是把几种状态信息存储在一个地方，让集群服务器的各个服务器都能访问到。

## Hadoop介绍

Hadoop是Apache旗下的一个用java语言实现开源软件框架，是一个开发和运行处理大规模数据的软件平台。允许使用简单的编程模型在大量计算机集群上对大型数据集进行分布式处理。

狭义上说，Hadoop指Apache这款开源框架，它的核心组件有：

**HDFS**（分布式文件系统）：解决海量数据存储

**YARN**（作业调度和集群资源管理的框架）：解决资源任务调度

**MAPREDUCE**（分布式运算编程框架）：解决海量数据计算

广义上来说，Hadoop通常是指一个更广泛的概念——Hadoop生态圈。



当下的Hadoop已经成长为一个庞大的体系，随着生态系统的成长，新出现的项目越来越多，其中不乏一些非Apache主管的项目，这些项目对HADOOP是很好的补充或者更高层的抽象。比如：

|  |  |
| --- | --- |
| **框架** | **用途** |
| HDFS | 分布式文件系统 |
| MapReduce | 分布式运算程序开发框架 |
| ZooKeeper | 分布式协调服务基础组件 |
| HIVE | 基于HADOOP的分布式数据仓库，提供基于SQL的查询数据操作 |
| FLUME | 日志数据采集框架 |
| oozie | 工作流调度框架 |
| Sqoop | 数据导入导出工具（比如用于mysql和HDFS之间） |
| Impala | 基于hive的实时sql查询分析 |
| Mahout | 基于mapreduce/spark/flink等分布式运算框架的机器学习算法库 |

## Hadoop发展简史

Hadoop是Apache Lucene创始人 Doug Cutting 创建的。最早起源于Nutch，它是Lucene的子项目。Nutch的设计目标是构建一个大型的全网搜索引擎，包括网页抓取、索引、查询等功能，但随着抓取网页数量的增加，遇到了严重的可扩展性问题：如何解决数十亿网页的存储和索引问题。

2003年Google发表了一篇论文为该问题提供了可行的解决方案。论文中描述的是谷歌的产品架构，该架构称为：谷歌分布式文件系统（GFS）,可以解决他们在网页爬取和索引过程中产生的超大文件的存储需求。

2004年 Google发表论文向全世界介绍了谷歌版的MapReduce系统。

同时期，以谷歌的论文为基础，Nutch的开发人员完成了相应的开源实现HDFS和MAPREDUCE，并从Nutch中剥离成为独立项目HADOOP，到2008年1月，HADOOP成为Apache顶级项目，迎来了它的快速发展期。

2006年Google发表了论文是关于BigTable的，这促使了后来的Hbase的发展。

因此，Hadoop及其生态圈的发展离不开Google的贡献。

## Hadoop特性优点

**扩容能力（Scalable）：**Hadoop是在可用的计算机集群间分配数据并完成计算任务的，这些集群可用方便的扩展到数以千计的节点中。

**成本低（Economical）**：Hadoop通过普通廉价的机器组成服务器集群来分发以及处理数据，以至于成本很低。

**高效率（Efficient）：**通过并发数据，Hadoop可以在节点之间动态并行的移动数据，使得速度非常快。

**可靠性（Rellable）：**能自动维护数据的多份复制，并且在任务失败后能自动地重新部署（redeploy）计算任务。所以Hadoop的按位存储和处理数据的能力值得人们信赖。

## Hadoop国内外应用

### Hadoop在国外应用的部分企业

**1.Yahoo**



Yahoo是Hadoop的最大支持者，Yahoo的Hadoop机器总节点数目已经超过42000个，有超过10万的核心CPU在运行Hadoop。最大的一个单Master节点集群有4500个节点（每个节点双路4核心CPUboxesw，4×1TB磁盘，16GBRAM）。总的集群存储容量大于350PB，每月提交的作业数目超过1000万个，在Pig中超过60%的Hadoop作业是使用Pig编写提交的。

Yahoo的Hadoop应用主要包括以下几个方面：

· 支持广告系统

· 用户行为分析

· 支持Web搜索

· 反垃圾邮件系统

· 个性化推荐

1. **Facebook**



主要用于存储内部日志的拷贝，作为一个源用于处理数据挖掘和日志统计。 主要使用了2个集群：一个由1100台节点组成的集群，包括8800核CPU（即每台机器8核）,和12000TB的原始存储(即每台机器12T硬盘) 一个有300台节点组成的集群，包括2400核CPU（即每台机器8核），和3000TB的原始存储(即每台机器12T硬盘) 由此基础上开发了基于SQL语法的项目：HIVE 。

Facebook使用Hadoop集群的机器节点超过1400台，共计11200个核心CPU，超过15PB原始存储容量，每个商用机器节点配置了8核CPU，12TB数据存储，主要使用StreamingAPI和JavaAPI编程接口。Facebook同时在Hadoop基础上建立了一个名为Hive的高级数据仓库框架，Hive已经正式成为基于Hadoop的Apache一级项目。

1. **IBM**



IBM蓝云也利用Hadoop来构建云基础设施。IBM蓝云使用的技术包括：Xen和PowerVM虚拟化的Linux操作系统映像及Hadoop并行工作量调度，并发布了自己的Hadoop发行版及大数据解决方案。

### Hadoop在国内应用的部分企业

1. **百度**



Hadoop集群规模达到近十个，单集群超过2800台机器节点，Hadoop机器总数有上万台机器，总的存储容量超过100PB，已经使用的超过74PB，每天提交的作业数目有数千个之多，每天的输入数据量已经超过7500TB，输出超过1700TB。

百度的Hadoop集群为整个公司的数据团队、大搜索团队、社区产品团队、广告团队，以及LBS团体提供统一的计算和存储服务，主要应用包括：

· 数据挖掘与分析

· 日志分析平台

· 数据仓库系统

· 推荐引擎系统

· 用户行为分析系统

1. **阿里巴巴**



阿里巴巴的Hadoop集群大约有3200台服务器，大约30?000物理CPU核心，总内存100TB，总的存储容量超过60PB，每天的作业数目超过150?000个，每天hivequery查询大于6000个，每天扫描数据量约为7.5PB，每天扫描文件数约为4亿，存储利用率大约为80%，CPU利用率平均为65%，峰值可以达到80%。

Hadoop集群拥有150个用户组、4500个集群用户，为电子商务网络平台提供底层的基础计算和存储服务，主要应用包括：

· 数据平台系统。

· 搜索支撑。

· 电子商务数据。

· 推荐引擎系统。

· 搜索排行榜。

1. **华为**



华为对Hadoop做出贡献的公司之一，排在Google和Cisco的前面，华为对Hadoop的HA方案，以及HBase领域有深入研究，并已经向业界推出了自己的基于Hadoop的大数据解决方案。

1. **腾讯**



TDW（Tencent distributed Data Warehouse，腾讯分布式数据仓库）基于开源软件Hadoop和Hive进行构建，打破了传统数据仓库不能线性扩展、可控性差的局限，并且根据腾讯数据量大、计算复杂等特定情况进行了大量优化和改造。

TDW服务覆盖了腾讯绝大部分业务产品，单集群规模达到4400台，CPU总核数达到10万左右，存储容量达到100PB；每日作业数100多万，每日计算量4PB，作业并发数2000左右；实际存储数据量80PB，文件数和块数达到6亿多；存储利用率83%左右，CPU利用率85%左右。经过四年多的持续投入和建设，TDW已经成为腾讯最大的离线数据处理平台。TDW的功能模块主要包括：Hive、MapReduce、HDFS、TDBank、Lhotse等

## Hadoop的历史版本和发行版公司

### Hadoop历史版本

1.x版本系列：hadoop版本当中的第二代开源版本，主要修复0.x版本的一些bug等，该版本已被淘汰

2.x版本系列：架构产生重大变化，引入了yarn平台等许多新特性,是现在使用的主流版本。

3.x版本系列: 加入多namenoode新特性，该版本是最新版本，但是还不太稳定。

### Hadoop发行版公司

Hadoop发行版本分为开源社区版 和 商业版。

社区版是指由Apache软件基金会维护的版本，是官方维护的版本体系。

商业版Hadoop是指由第三方商业公司在社区版Hadoop基础上进行了一些修改、整合以及各个服务组件兼容性测试而发行的版本，比较著名的有cloudera的CDH、mapR、hortonWorks等。

**免费开源版本Apache:**

[**http://hadoop.apache.org/**](http://hadoop.apache.org/)

优点：拥有全世界的开源贡献者，代码更新迭代版本比较快，

缺点：版本的升级，版本的维护，版本的兼容性，版本的补丁都可能考虑不太周到，

apache所有软件的下载地址（包括各种历史版本）：

[**http://archive.apache.org/dist/**](http://archive.apache.org/dist/)

**免费开源版本HortonWorks：**

[**http://hadoop.apache.org/**](http://hadoop.apache.org/)

hortonworks主要是雅虎主导Hadoop开发的副总裁，带领二十几个核心成员成立Hortonworks，核心产品软件HDP（ambari），HDF免费开源，并且提供一整套的web管理界面，供我们可以通过web界面管理我们的集群状态，web管理界面软件HDF网址（**<http://ambari.apache.org/>**），2018年，大数据领域的两大巨头公司Cloudera和Hortonworks宣布平等合并，Cloudera以股票方式收购Hortonworks，Cloudera股东最终获得合并公司60%的股份

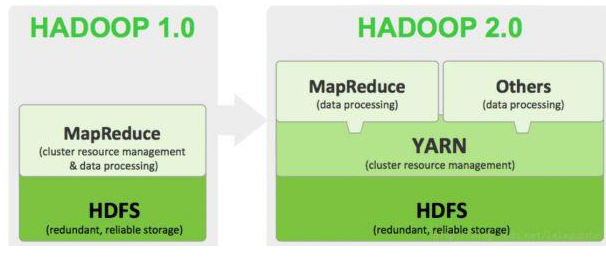
**软件收费版本Cloudera:**

[**https://www.cloudera.com/**](https://www.cloudera.com/)

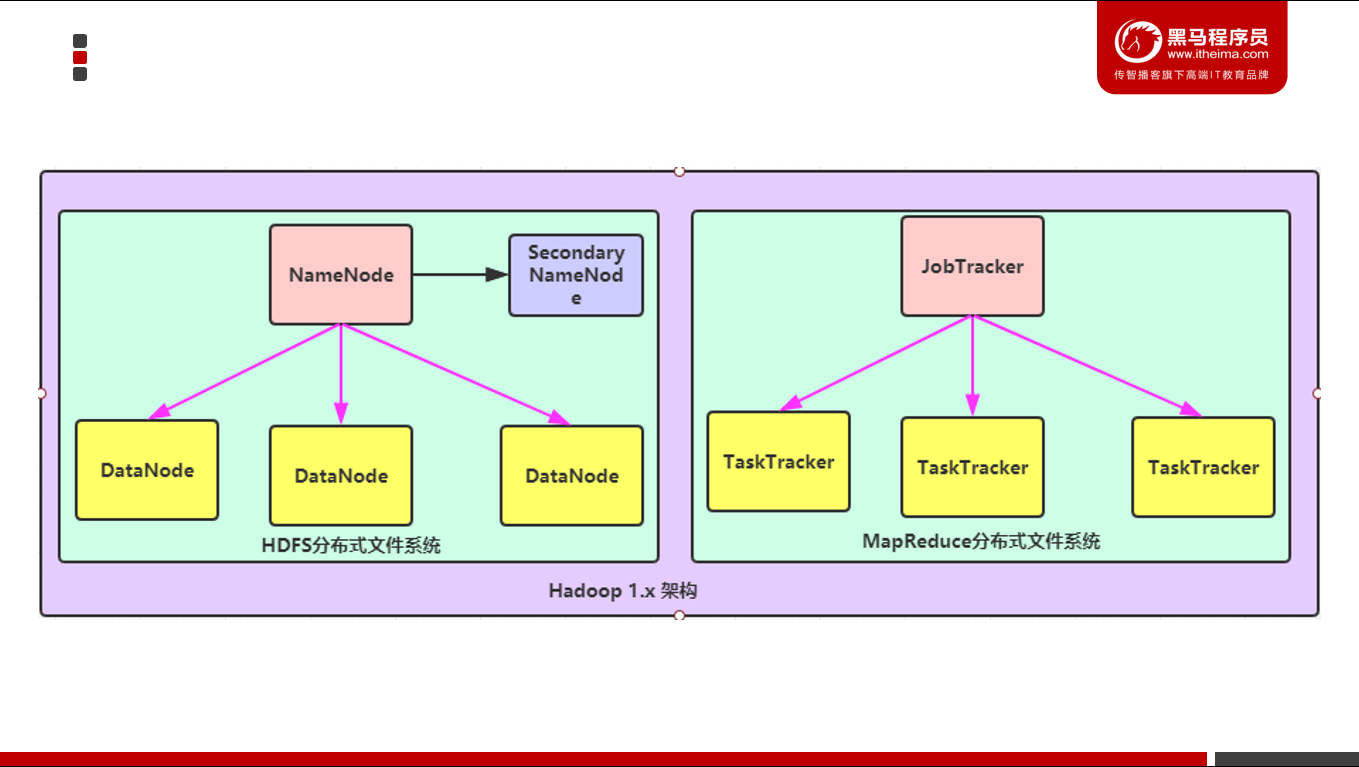
cloudera主要是美国一家大数据公司在apache开源hadoop的版本上，通过自己公司内部的各种补丁，实现版本之间的稳定运行，大数据生态圈的各个版本的软件都提供了对应的版本，解决了版本的升级困难，版本兼容性等各种问题

在本课程中使用的是Apache版的Hadoop，版本号为2.7.5。

## Hadoop架构



### 1.x的版本架构模型介绍



**文件系统核心模块：**

NameNode：集群当中的主节点，管理元数据(文件的大小，文件的位置，文件的权限)，主要用于管理集群当中的各种数据

SecondaryNameNode：主要能用于hadoop当中元数据信息的辅助管理

DataNode：集群当中的从节点，主要用于存储集群当中的各种数据

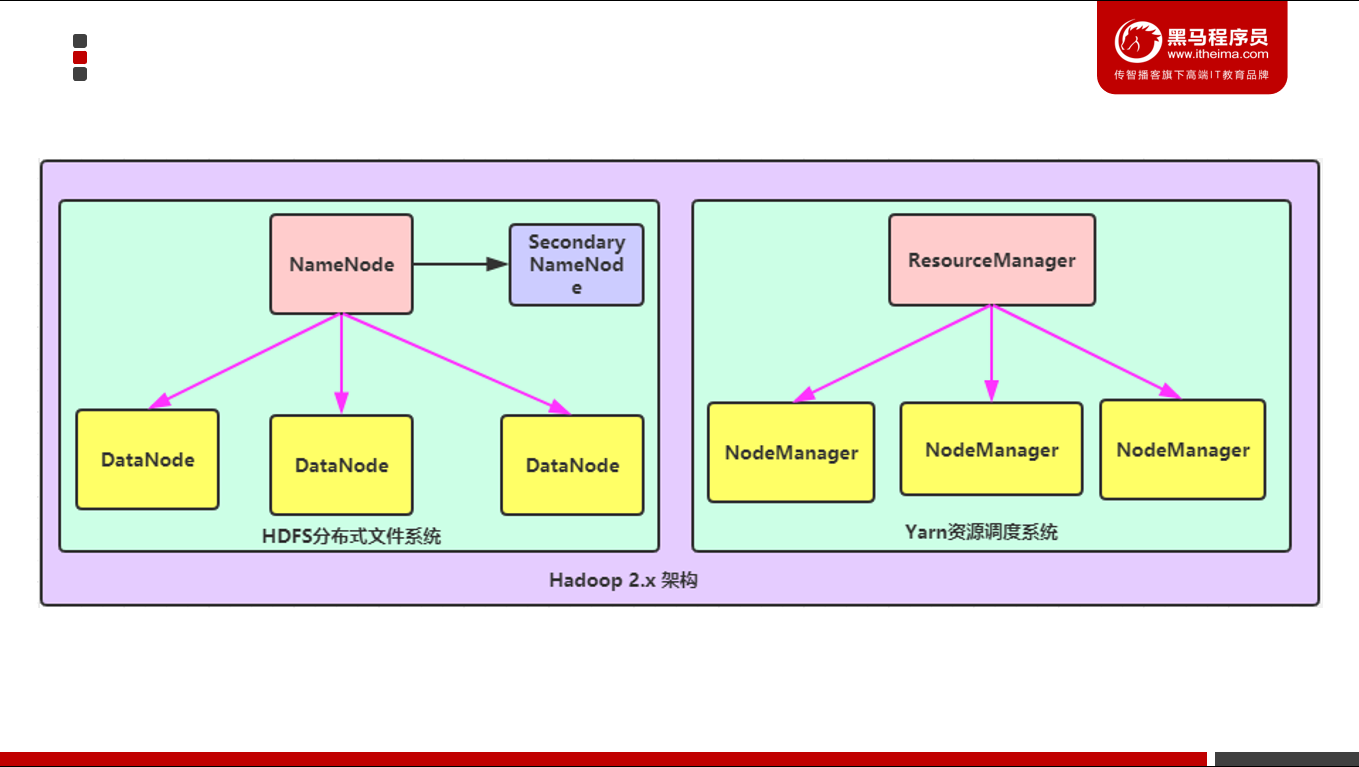
**数据计算核心模块：**

JobTracker：接收用户的计算请求任务，并分配任务给从节点

TaskTracker：负责执行主节点JobTracker分配的任务

### 2.x的版本架构模型介绍

* **第一种：NameNode与ResourceManager单节点架构模型**



**文件系统核心模块：**

NameNode：集群当中的主节点，主要用于管理集群当中的各种数据

secondaryNameNode：主要能用于hadoop当中元数据信息的辅助管理

DataNode：集群当中的从节点，主要用于存储集群当中的各种数据

**数据计算核心模块：**

ResourceManager：接收用户的计算请求任务，并负责集群的资源分配

NodeManager：负责执行主节点APPmaster分配的任务

* **第二种：NameNode高可用与ResourceManager单节点架构模型**



**文件系统核心模块：**

NameNode：集群当中的主节点，主要用于管理集群当中的各种数据，其中NameNode可以有两个，形成高可用状态

DataNode：集群当中的从节点，主要用于存储集群当中的各种数据

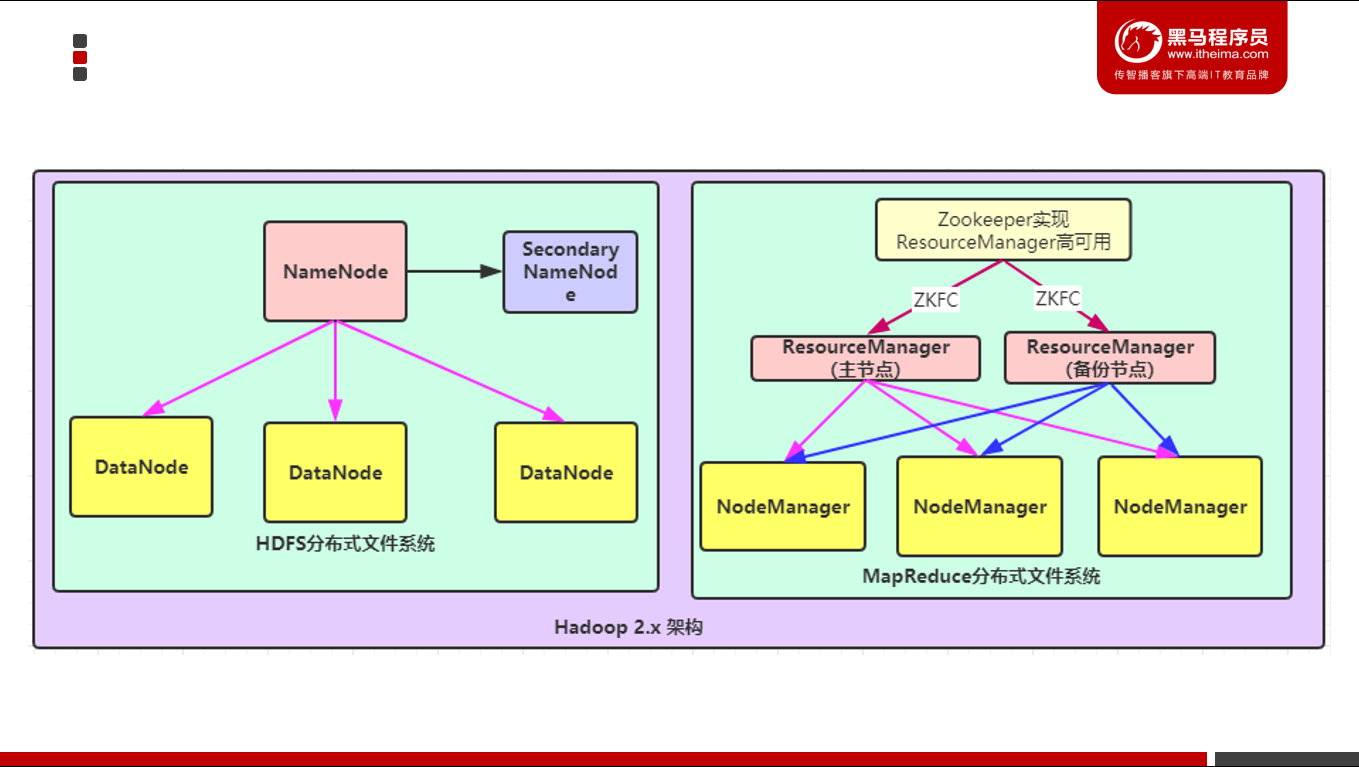
JournalNode：文件系统元数据信息管理

**数据计算核心模块：**

ResourceManager：接收用户的计算请求任务，并负责集群的资源分配，以及计算任务的划分

NodeManager：负责执行主节点ResourceManager分配的任务

* **第三种：NameNode单节点与ResourceManager高可用架构模型**



**文件系统核心模块：**

NameNode：集群当中的主节点，主要用于管理集群当中的各种数据

secondaryNameNode：主要能用于hadoop当中元数据信息的辅助管理

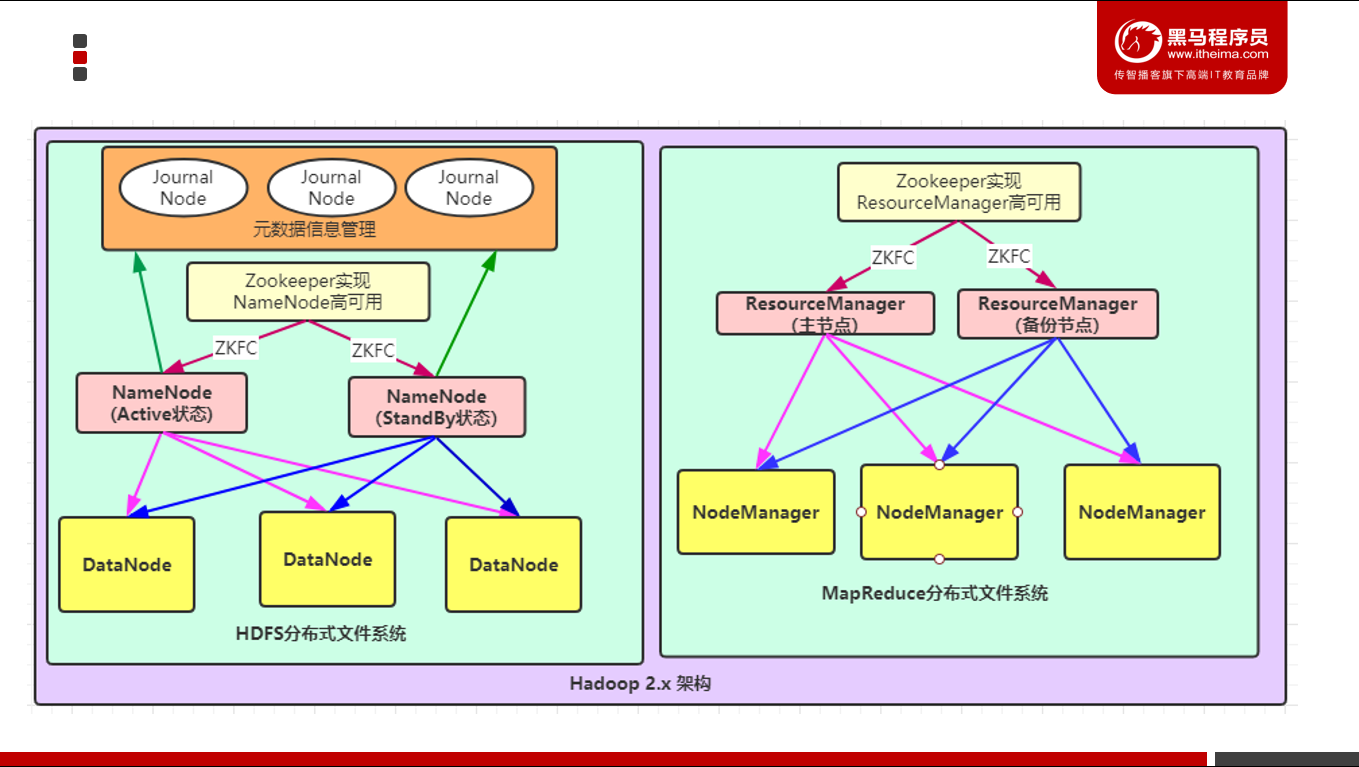
DataNode：集群当中的从节点，主要用于存储集群当中的各种数据

**数据计算核心模块：**

ResourceManager：接收用户的计算请求任务，并负责集群的资源分配，以及计算任务的划分，通过zookeeper实现ResourceManager的高可用

NodeManager：负责执行主节点ResourceManager分配的任务

* **第四种：NameNode与ResourceManager高可用架构模型**



**文件系统核心模块：**

NameNode：集群当中的主节点，主要用于管理集群当中的各种数据，一般都是使用两个，实现HA高可用

JournalNode：元数据信息管理进程，一般都是奇数个

DataNode：从节点，用于数据的存储

**数据计算核心模块：**

ResourceManager：Yarn平台的主节点，主要用于接收各种任务，通过两个，构建成高可用

NodeManager：Yarn平台的从节点，主要用于处理ResourceManager分配的任务

### 3.x的版本架构和模型介绍

**介绍**

    由于Hadoop 2.0是基于JDK 1.7开发的，而JDK 1.7在2015年4月已停止更新，这直接迫使Hadoop社区基于JDK 1.8重新发布一个新的Hadoop版本，即hadoop 3.0。Hadoop 3.0中引入了一些重要的功能和优化，包括HDFS 可擦除编码、多Namenode支持、MR Native Task优化、YARN基于cgroup的内存和磁盘IO隔离、YARN container resizing等。

    Apache hadoop 项目组最新消息，hadoop3.x以后将会调整方案架构，将Mapreduce 基于内存+io+磁盘，共同处理数据。

改变最大的是hdfs,hdfs 通过最近block块计算，根据最近计算原则，本地block块，加入到内存，先计算，通过IO，共享内存计算区域，最后快速形成计算结果，比Spark快10倍。

**Hadoop 3.0新特性**

Hadoop 3.0在功能和性能方面，对hadoop内核进行了多项重大改进，主要包括：

**通用性**

1. 精简Hadoop内核，包括剔除过期的API和实现，将默认组件实现替换成最高效的实现。
2. Classpath isolation：以防止不同版本jar包冲突
3. Shell脚本重构： Hadoop 3.0对Hadoop的管理脚本进行了重构，修复了大量bug，增加了新特性。

**HDFS**

Hadoop3.x中Hdfs在可靠性和支持能力上作出很大改观：

1.HDFS支持数据的擦除编码，这使得HDFS在不降低可靠性的前提下，节省一半存储空间。

2.多NameNode支持，即支持一个集群中，一个active、多个standby namenode部署方式。注：多ResourceManager特性在hadoop 2.0中已经支持。

**MapReduce**

Hadoop3.X中的MapReduce较之前的版本作出以下更改：

1.Tasknative优化：为MapReduce增加了C/C++的map output collector实现（包括Spill，Sort和IFile等），通过作业级别参数调整就可切换到该实现上。对于shuffle密集型应用，其性能可提高约30%。

2.MapReduce内存参数自动推断。在Hadoop 2.0中，为MapReduce作业设置内存参数非常繁琐，一旦设置不合理，则会使得内存资源浪费严重，在Hadoop3.0中避免了这种情况。

**HDFS纠删码**

在Hadoop3.X中，HDFS实现了Erasure Coding这个新功能。Erasure coding纠删码技术简称EC，是一种数据保护技术.最早用于通信行业中数据传输中的数据恢复，是一种编码容错技术。

它通过在原始数据中加入新的校验数据，使得各个部分的数据产生关联性。在一定范围的数据出错情况下，通过纠删码技术都可以进行恢复。

hadoop-3.0之前，HDFS存储方式为每一份数据存储3份，这也使得存储利用率仅为1/3，hadoop-3.0引入纠删码技术(EC技术)，实现1份数据+0.5份冗余校验数据存储方式。

与副本相比纠删码是一种更节省空间的数据持久化存储方法。标准编码(比如Reed-Solomon(10,4))会有1.4 倍的空间开销；然而HDFS副本则会有3倍的空间开销。

**MapReduce优化**

Hadoop3.x中的MapReduce添加了Map输出collector的本地实现，对于shuffle密集型的作业来说，这将会有30%以上的性能提升。

**支持多个NameNodes**

     最初的HDFS NameNode high-availability实现仅仅提供了一个active NameNode和一个Standby NameNode；并且通过将编辑日志复制到三个JournalNodes上，这种架构能够容忍系统中的任何一个节点的失败。

    然而，一些部署需要更高的容错度。我们可以通过这个新特性来实现，其允许用户运行多个Standby NameNode。比如通过配置三个NameNode和五个JournalNodes，这个系统可以容忍2个节点的故障，而不是仅仅一个节点。

**默认端口更改**

    在hadoop3.x之前，多个Hadoop服务的默认端口都属于Linux的临时端口范围（32768-61000）。这就意味着用户的服务在启动的时候可能因为和其他应用程序产生端口冲突而无法启动。

    现在这些可能会产生冲突的端口已经不再属于临时端口的范围，这些端口的改变会影响NameNode, Secondary NameNode, DataNode以及KMS。与此同时，官方文档也进行了相应的改变，具体可以参见 HDFS-9427以及HADOOP-12811。

Namenode ports: 50470 --> 9871, 50070--> 9870, 8020 --> 9820

Secondary NN ports: 50091 --> 9869,50090 --> 9868

Datanode ports: 50020 --> 9867, 50010--> 9866, 50475 --> 9865, 50075 --> 9864

Kms servers ports: 16000 --> 9600 (原先的16000与HMaster端口冲突)

**YARN 资源类型**

    YARN 资源模型（YARN resource model）已被推广为支持用户自定义的可数资源类型（support user-defined countable resource types），不仅仅支持 CPU 和内存。

    比如集群管理员可以定义诸如 GPUs、软件许可证（software licenses）或本地附加存储器（locally-attached storage）之类的资源。YARN 任务可以根据这些资源的可用性进行调度。

## Hadoop集群搭建

### 集群简介

HADOOP集群具体来说包含两个集群：HDFS集群和YARN集群，两者逻辑上分离，但物理上常在一起。

**HDFS集群负责海量数据的存储，**集群中的角色主要有：

NameNode、DataNode、SecondaryNameNode

**YARN集群负责海量数据运算时的资源调度**，集群中的角色主要有：

ResourceManager、NodeManager

那mapreduce是什么呢？它其实是一个分布式运算编程框架，是应用程序开发包，由用户按照编程规范进行程序开发，后打包运行在HDFS集群上，并且受到YARN集群的资源调度管理。

### 集群部署方式

**Hadoop部署方式分三种：**

**1、Standalone mode（独立模式）**

独立模式又称为单机模式，仅1个机器运行1个java进程，主要用于调试。

**2、Pseudo-Distributed mode（伪分布式模式）**

伪分布模式也是在1个机器上运行HDFS的NameNode和DataNode、YARN的 ResourceManger和NodeManager，但分别启动单独的java进程，主要用于调试。

**3、Cluster mode（群集模式）**

集群模式主要用于生产环境部署。会使用N台主机组成一个Hadoop集群。这种部署模式下，主节点和从节点会分开部署在不同的机器上。

本课程搭建的是集群模式，以三台主机为例，以下是集群规划:

|  |  |  |  |
| --- | --- | --- | --- |
| 主机  组件 | node1  (192.168.88.161) | node2  (192.168.88.162) | node3  (192.168.88.163) |
| NameNode | 是 | 否 | 否 |
| Secondary  Namenode | 否 | 是 | 否 |
| DataNode | 是 | 是 | 是 |
| ResourceManager | 是 | 否 | 否 |
| NodeManager | 是 | 是 | 是 |

### 集群环境准备

#### 服务器准备

本案例使用VMware Workstation Pro虚拟机创建虚拟服务器来搭建HADOOP集群，所用软件及版本如下：

VMware Workstation Pro 15.0

Centos7.7-64bit

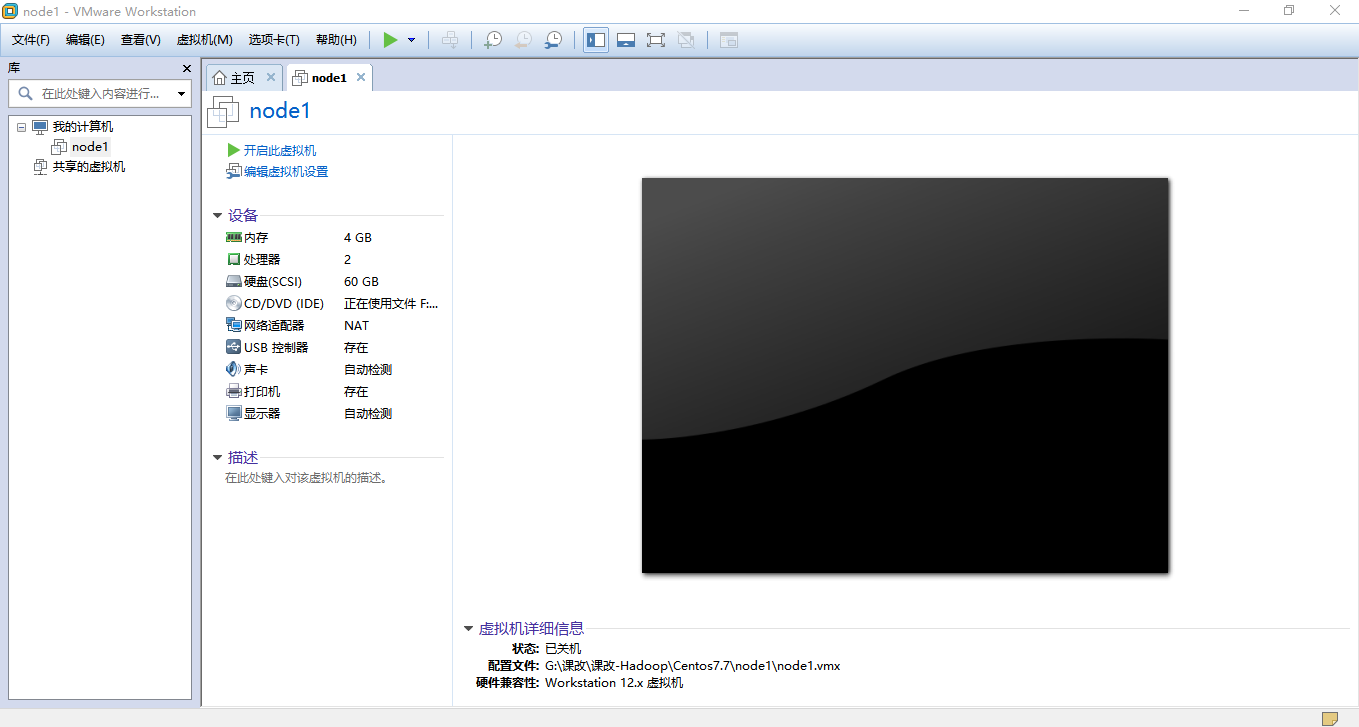
#### 三台虚拟机创建

**第一种方式：通过iso镜像文件来进行安装(不推荐)**

**第二种方式：直接复制安装好的虚拟机文件（强烈推荐）**

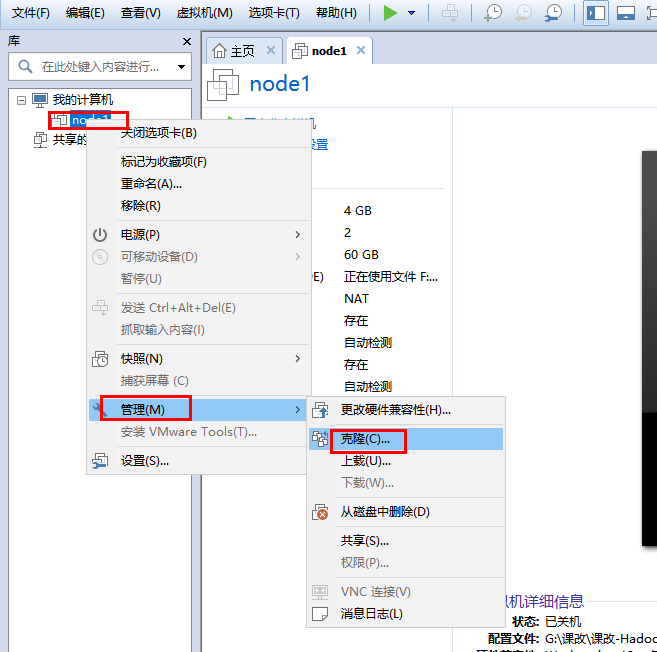
在课程资料里边已经提供了一个安装好的虚拟机node1，我们需要根据这个虚拟机克隆出另外两台虚拟机出来,步骤如下:

1. 使用VMware加载资料中虚拟机node1



2）克隆第二台虚拟机，注意克隆虚拟机的时候，虚拟机必须是关闭状态

右键点击node1



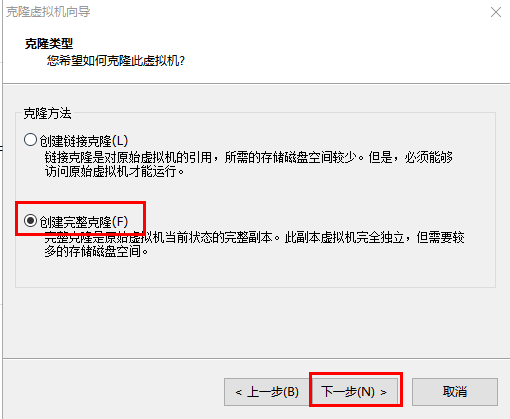
下一步



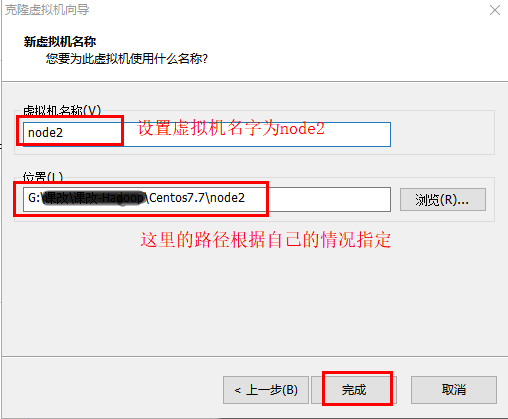
下一步



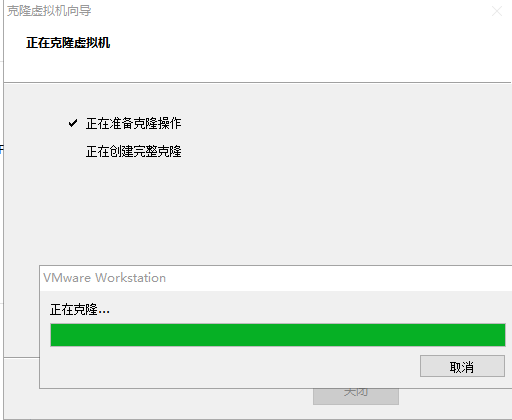
创建完整克隆，下一步



指定虚拟机名字和存放位置,三台虚拟机的存放路径尽量在一起，不在一起也没关系



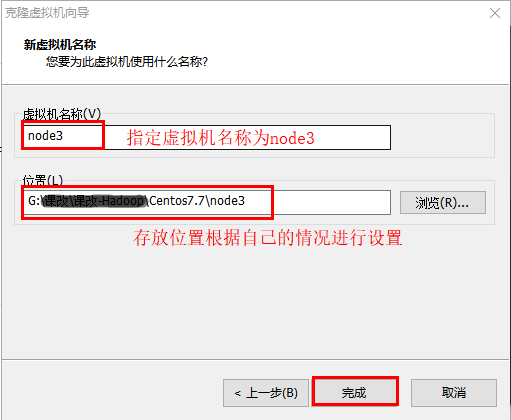
等待克隆完毕



关闭



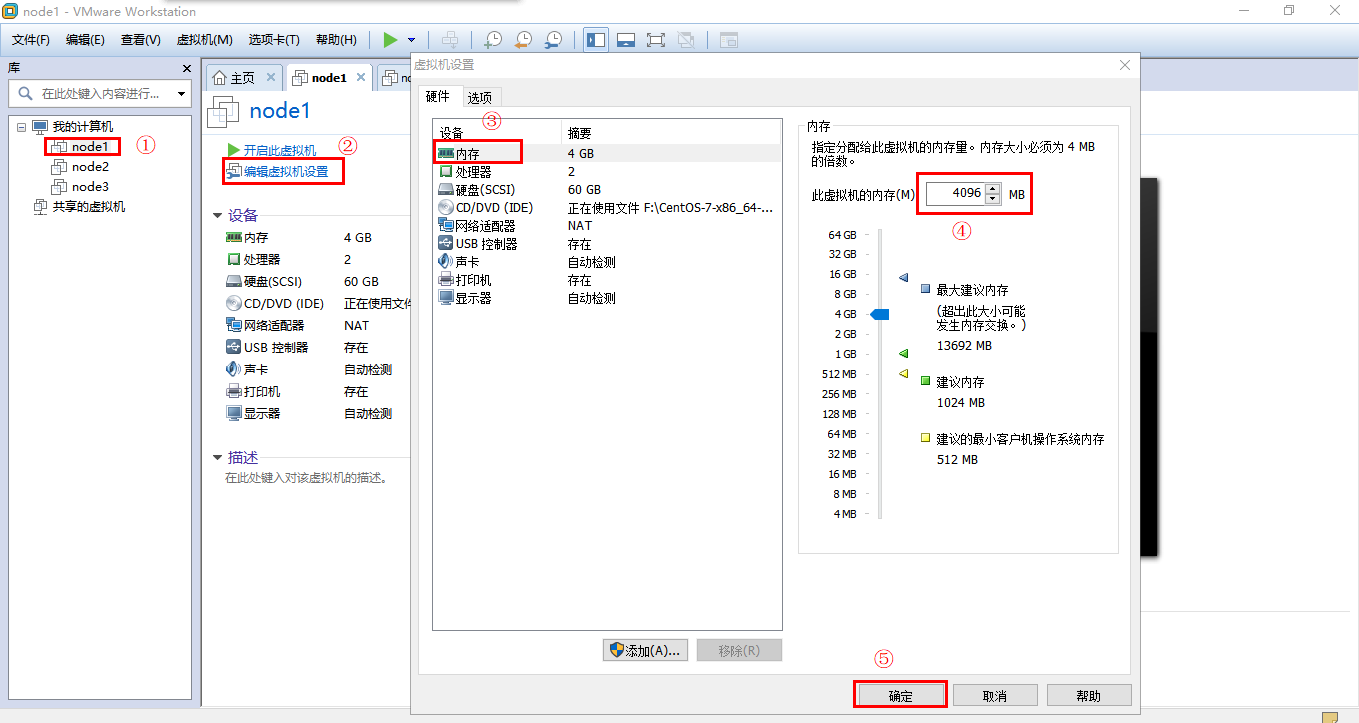
1. 克隆第三台虚拟机，通过node1虚拟机克隆，克隆方式是一样的，注意修改虚拟机的名称和存放位置。



#### 设置三台虚拟机的内存

三台虚拟机再加上windows本身, 需要同时运行4台机器, 所以在分配的时候,, 每台虚拟机的内存为: 总内存 ÷ 4，比如电脑总内存为16G,则每台虚拟机内存为4G。

下面是以node1为例对内存进行配置:



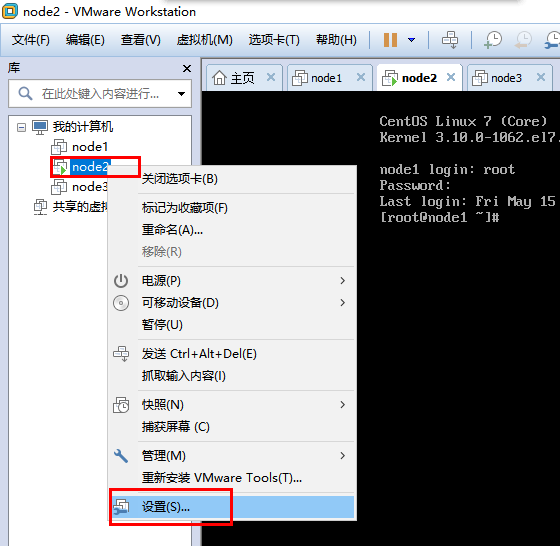
#### 配置MAC地址

node2和node3都是从node1克隆过来的，他们的MAC地址都一样，所以需要让node2和node3重新生成MAC地址，生成方式如下:

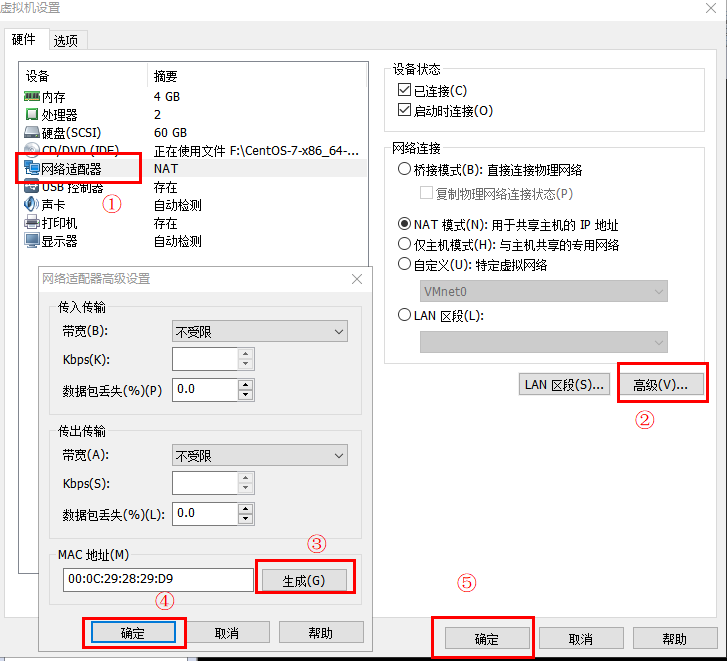
**1、配置node2的MAC地址**

1）使用VMware打开node2

2）右键点击node2，选择设置



3)生成新的MAC地址



1. **配置node3的MAC地址**

**node3的配置方式和node2相同，不再重复**

#### 配置IP地址

三台虚拟机的IP地址配置如下：

node1: 192.168.88.161

node2 192.168.88.162

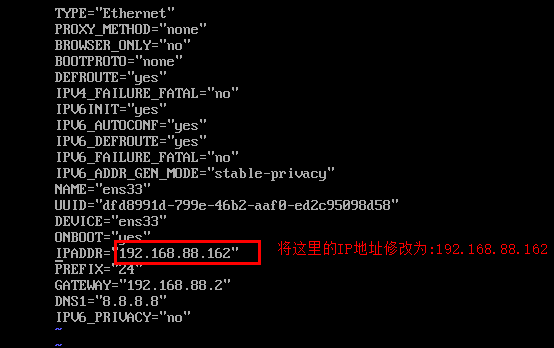
node3: 192.168.88.163

node1的IP地址已经配置好了，接下来配置node2和node3：

**1：配置node2主机IP**

1. 修改ip配置文件,设置IP地址

|  |
| --- |
| **vim** **/**etc**/**sysconfig**/**network-scripts**/**ifcfg-ens33 |



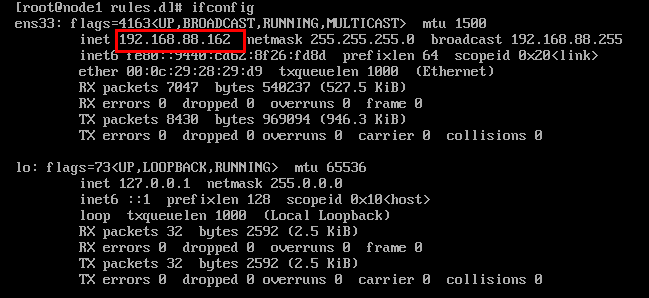
1. **重启网络服务**

|  |
| --- |
| **service** network restart |



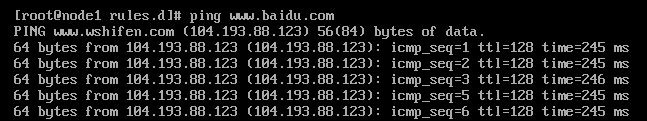
1. **查看ip地址**

|  |
| --- |
| ifconfig |



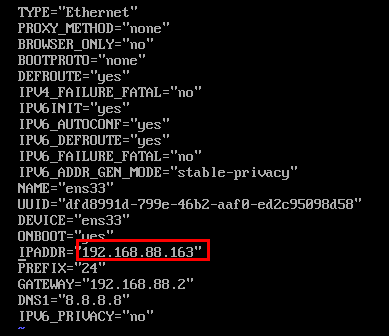
1. 测试网络连接

|  |
| --- |
| **ping** www.baidu.com |



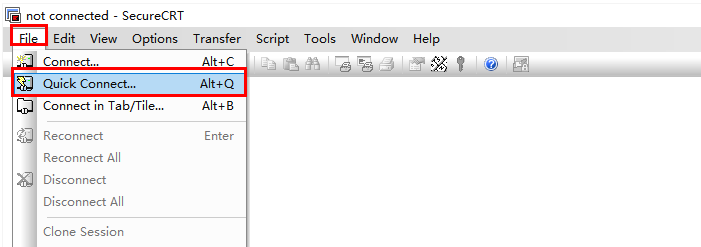
**2：配置node3主机IP**

**node3主机IP的配置方式和node2一样，将其IP地址设置为:192.168.88.163，在这里不再重复。**

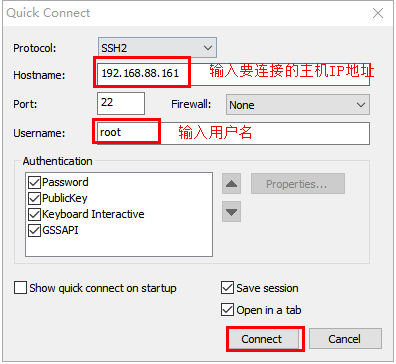


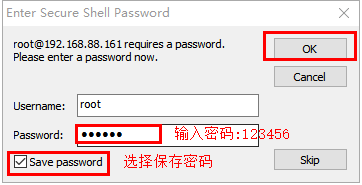
#### 使用CRT连接三台虚拟机

1、建立连接

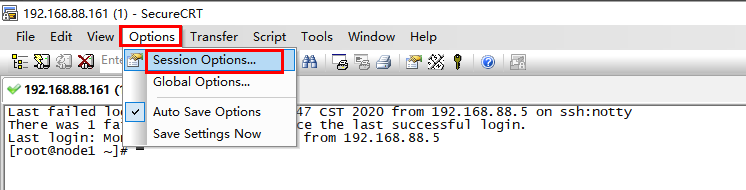


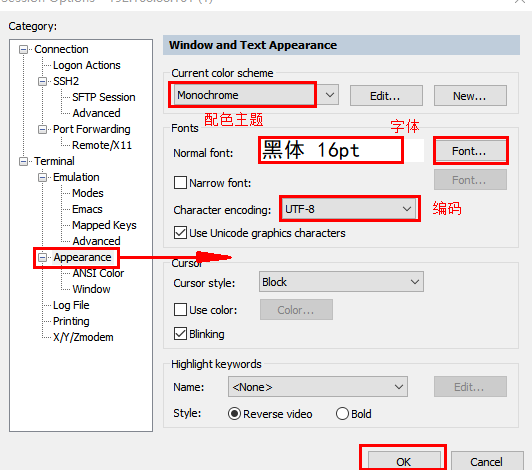
2、参数配置

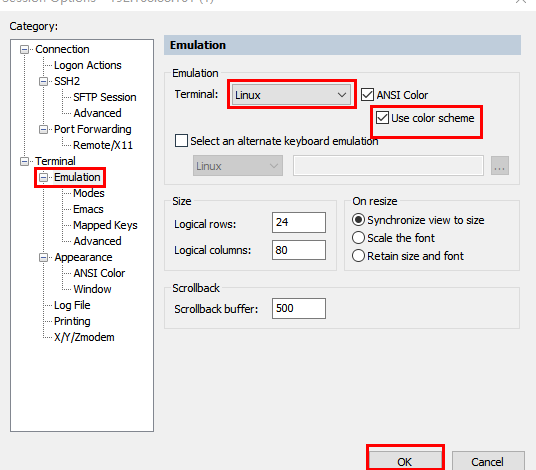




3、设置主题，颜色和仿真







#### 设置主机名和域名映射

1. **配置每台虚拟机主机名:**

分别编辑每台虚拟机的hostname文件，直接填写主机名，保存退出即可。

|  |
| --- |
| vim /etc/hostname |

第一台主机主机名为:node1

第二台主机主机名为:node2

第三台主机主机名为:node3

1. **配置每台虚拟机域名映射**

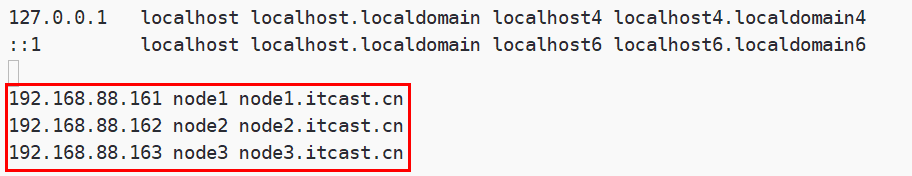
分别编辑每台虚拟机的hosts文件，在原有内容的基础上，填下以下内容:

注意：不要修改文件原来的内容，三台虚拟机的配置内容都一样。

|  |
| --- |
| vim /etc/hosts |

|  |
| --- |
| 192.168.88.161 node1 node1.itcast.cn  192.168.88.162 node2 node2.itcast.cn  192.168.88.163 node3 node3.itcast.cn |

配置后效果如下:



#### 关闭三台虚拟机的防火墙和Selinux

1. **关闭每台虚拟机的防火墙**

在每台虚拟机上分别执行以下指令:

|  |
| --- |
| **systemctl** stop firewalld.service #停止firewall  **systemctl** disable firewalld.service #禁止firewall开机启动 |

关闭之后,查看防火墙状态:

|  |
| --- |
| **systemctl** status firewalld.service |



1. **关闭每台虚拟机的Selinux**

**1）什么是SELinux ?**

1)SELinux是Linux的一种安全子系统

2)Linux中的权限管理是针对于文件的, 而不是针对进程的, 也就是说, 如果root启动了某个进程, 则这个进程可以操作任何一个文件。

3）SELinux在Linux的文件权限之外, 增加了对进程的限制, 进程只能在进程允许的范围内操作资源

**2）为什么要关闭SELinux**

如果开启了SELinux, 需要做非常复杂的配置, 才能正常使用系统, 在学习阶段, 在非生产环境, 一般不使用SELinux

SELinux的工作模式：

**enforcing 强制模式**

**permissive 宽容模式**

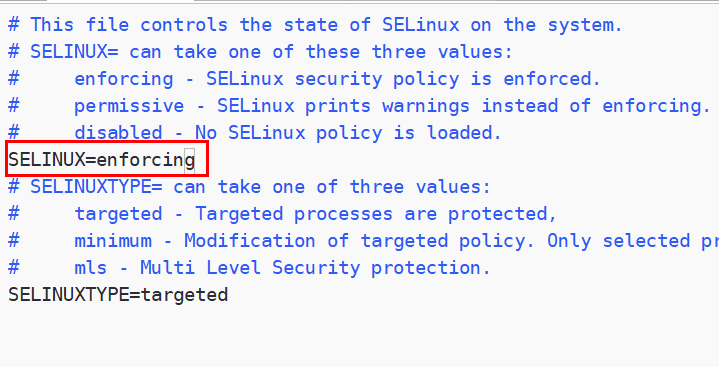
**disabled 关闭**

1. **关闭SELinux方式**

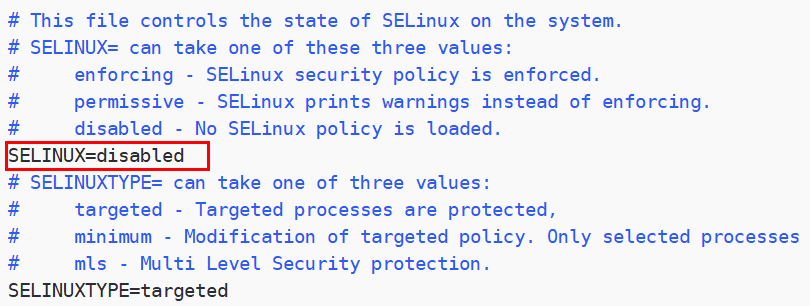
编辑每台虚拟机的Selinux的配置文件

|  |
| --- |
| **vim** **/**etc**/**selinux**/**config |

**Selinux的默认工作模式是强制模式，配置如下：**

****

**将Selinux工作模式关闭:**



1. **分别重启三台虚拟机**

|  |
| --- |
| reboot |

#### 三台机器机器免密码登录

**1、为什么要免密登录**

Hadoop 节点众多, 所以一般在主节点启动从节点, 这个时候就需要程序自动在主节点登录到从节点中, 如果不能免密就每次都要输入密码, 非常麻烦。



**2、免密 SSH 登录的原理**

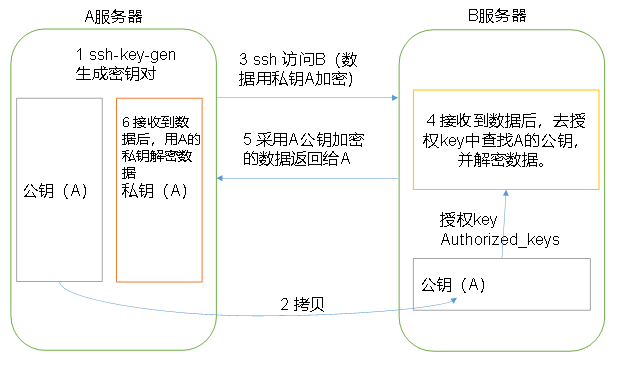
1. 需要先在 B节点 配置 A节点 的公钥

2. A节点 请求 B节点 要求登录

3. B节点 使用 A节点 的公钥, 加密一段随机文本

4. A节点 使用私钥解密, 并发回给 B节点

5. B节点 验证文本是否正确



**3、实现步骤**

**第一步：三台机器生成公钥与私钥**

在三台机器执行以下命令，生成公钥与私钥

|  |
| --- |
| ssh-keygen -t rsa |

执行该命令之后，按下三个回车即可，然后敲（三个回车），就会生成两个文件id\_rsa（私钥）、id\_rsa.pub（公钥），默认保存在/root/.ssh目录。



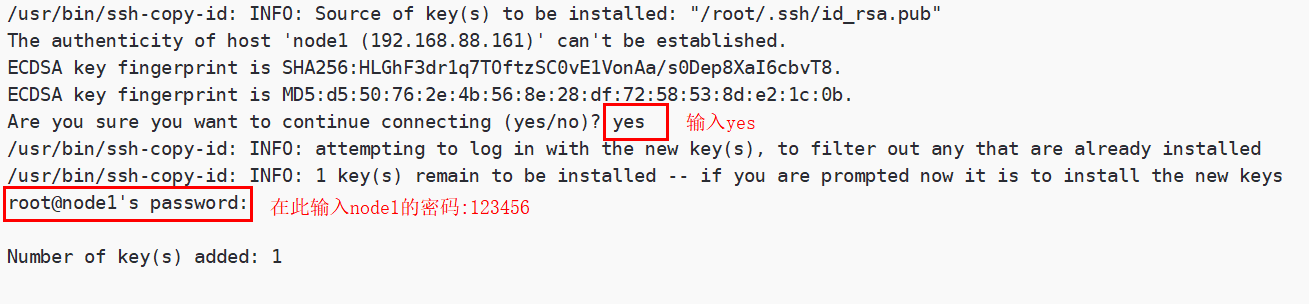
**第二步：拷贝公钥到同一台机器**

三台机器将拷贝公钥到第一台机器

三台机器执行命令：

|  |
| --- |
| ssh-copy-id node1 |

在执行该命令之后，需要输入yes和node1的密码:



**第三步:复制第一台机器的认证到其他机器**

将第一台机器的公钥拷贝到其他机器上

在第一台机器上指行以下命令

|  |
| --- |
| scp **/**root**/.**ssh**/**authorized\_keys node2**:/**root**/.**ssh  scp **/**root**/.**ssh**/**authorized\_keys node3**:/**root**/.**ssh |

执行命令时，需要输入yes和对方的密码

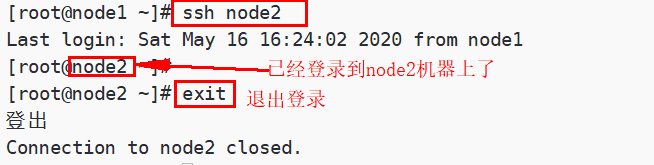
**第三步:测试SSH免密登录**

可以在任何一台主机上通过ssh 主机名命令去远程登录到该主机，输入exit退出登录

例如：在node1机器上，免密登录到node2机器上

|  |
| --- |
| ssh node1  exit |

执行效果如下：



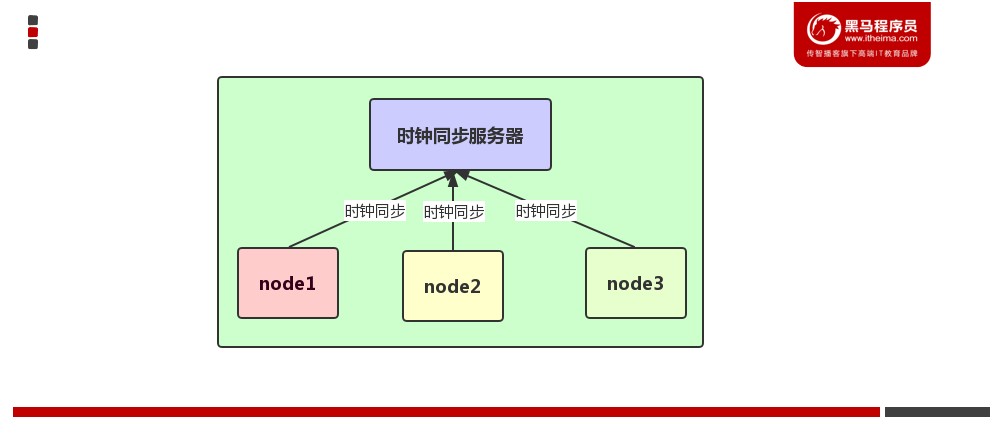
#### 三台机器时钟同步

**为什么需要时间同步**

因为很多分布式系统是有状态的, 比如说存储一个数据, A节点 记录的时间是1, B节点 记录的时间是2, 就会出问题

**时钟同步方式**

方式一：通过网络进行时钟同步



通过网络连接外网进行时钟同步,必须保证虚拟机连上外网

1. 在线安装ntp

NTP是网络时间协议(Network Time Protocol)，它是用来同步网络中各个计算机的时间的协议。

|  |
| --- |
| **yum** install **-**y ntp |

1. 启动定时任务

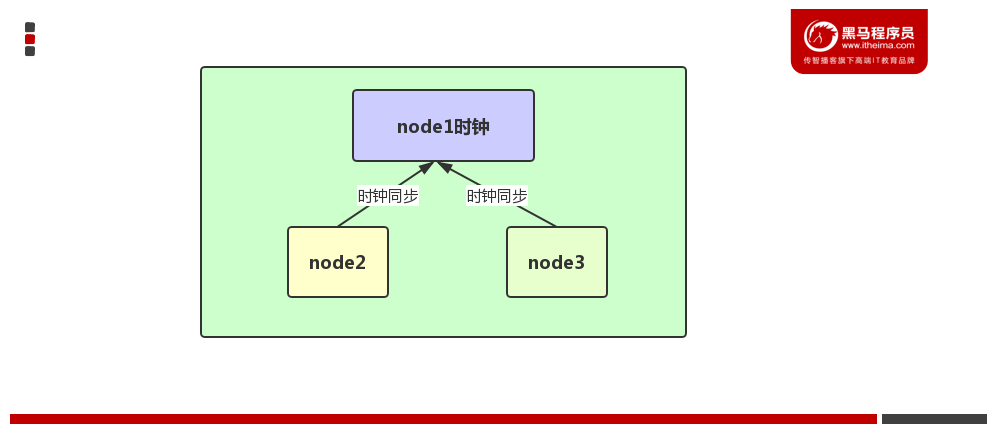
|  |
| --- |
| **crontab** -e |

随后在输入界面键入以下内容，每隔一分钟就去连接阿里云时间同步服务器，进行时钟同步

|  |
| --- |
| **\*/**1 **\*** **\*** **\*** **\*** **/**usr**/**sbin**/**ntpdate ntp4.aliyun.com**;** |

注意，在业务环境中，为了安全, 一般情况下，大数据集群的节点不允许连接外网，所以这种方式只能用在测试环境中。

方式二：通过某一台机器进行同步



以192.168.88.161这台服务器的时间为准进行时钟同步

##### 第一步:在node1虚拟机安装ntp并启动

安装ntp服务

|  |
| --- |
| **yum** **-**y install ntp |

启动ntp服务

|  |
| --- |
| **systemctl** start ntpd |

设置ntpd的服务开机启动

|  |
| --- |
| #关闭chrony,Chrony是NTP的另一种实现  **systemctl** disable chrony  #设置ntp服务为开机启动  **systemctl** enable ntpd |

##### 第二步:编辑node1的/etc/ntp.conf文件

编辑node1机器的/etc/ntp.conf

|  |
| --- |
| **vim** **/**etc**/**ntp.conf |

在文件中添加如下内容(授权192.168.88.0-192.168.88.255网段上的所有机器可以从这台机器上查询和同步时间)

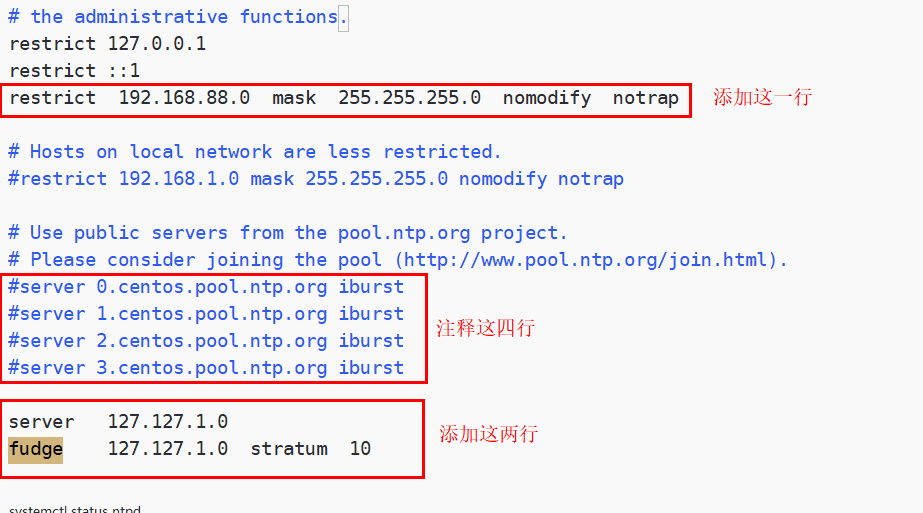
|  |
| --- |
| restrict 192**.**168**.**88**.**0 mask 255**.**255**.**255**.**0 nomodify notrap |

注释一下四行内容:(集群在局域网中，不使用其他互联网上的时间)

|  |
| --- |
| #servers 0.centos.pool.ntp.org  #servers 1.centos.pool.ntp.org  #servers 2.centos.pool.ntp.org  #servers 3.centos.pool.ntp.org |

去掉以下内容的注释，如果没有这两行注释，那就自己添加上(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)

|  |
| --- |
| servers 127.127.1.0  fudge 127.127.1.0 stratum 10 |

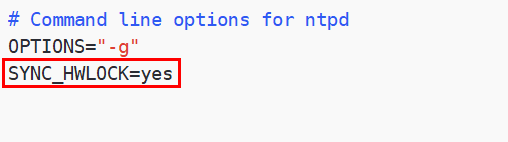


配置以下内容，保证BIOS与系统时间同步

|  |
| --- |
| **vim** **/**etc**/**sysconfig**/**ntpd |

添加一行内容

|  |
| --- |
| SYNC\_HWLOCK=yes |



##### 第三步：另外两台机器与第一台机器时间同步

另外两台机器与192.168.88.161进行时钟同步，在node2和node3机器上分别进行以下操作

|  |
| --- |
| crontab -e |

添加以下内容:(每隔一分钟与node1进行时钟同步)

|  |
| --- |
| \*/1 \* \* \* \* /usr/sbin/ntpdate 192.168.88.161 |

#### JDK环境安装

三台机器安装jdk

1. 在三台机器上分别创建安装目录

|  |
| --- |
| **mkdir** -p **/**export**/**software #软件包存放目录  **mkdir** -p **/export/server** #安装目录  **mkdir** -p **/**export**/**data #数据存放目录 |

2、在三台虚拟机上查看自带的openjdk并卸载

|  |
| --- |
| **rpm** **-**qa **|** **grep** openjdk  **rpm** -e java-1.7.0-openjdk-headless-1.7.0.221-2.6.18.1.el7.x86\_64 java-1.7.0-openjdk-1.7.0.221-2.6.18.1.el7.x86\_64 **--**nodeps |

3、在node1机上传安装包并解压

上传jdk到node1的/export/software路径下去并解压，在这里使用rz命令上传,如果不能使用，则需要执行命令:yum -y install lrzsz命令来在线安装lrzsz，-y参数表示安装过程提示选择全部为"yes"。

|  |
| --- |
| **tar** **-**zxvf jdk-8u241-linux-x64.tar.gz -C /export/servers/ |

4、在node1配置环境变量

|  |
| --- |
| **vim** **/**etc**/**profile |

添加如下内容

|  |
| --- |
| **export** JAVA\_HOME**=/export/servers/**jdk1.8.0\_241  **export** PATH**=:$JAVA\_HOME/**bin**:$PATH** |

1. 在node1上进行文件分发

远程拷贝jdk安装目录，将其拷贝到node2和node3主机

|  |
| --- |
| scp -r **/export/servers/**jdk1.8.0\_241**/** node2**:/export/servers/**  scp -r **/export/servers/**jdk1.8.0\_241**/** node3**:/export/servers/** |

远程拷贝/etc/profile配置文件，将其拷贝到node2和node3主机

|  |
| --- |
| scp **/**etc**/**profile node2**:/**etc**/**  scp **/**etc**/**profile node3**:/**etc**/** |

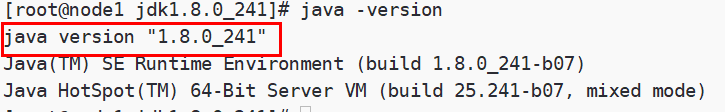
修改完成之后记得在三台主机上分别执行source /etc/profile使配置生效

|  |
| --- |
| source **/**etc**/**profile |

5、测试

在三台主机上执行以下指令，测试是否安装成功:

|  |
| --- |
| java -version |



**扩展资料，**

三台虚拟机一键安装JDK脚本：insall\_jdk.sh,

运行脚本之前，需要提前将jdk的安装包上传到node1的/export/software目录。

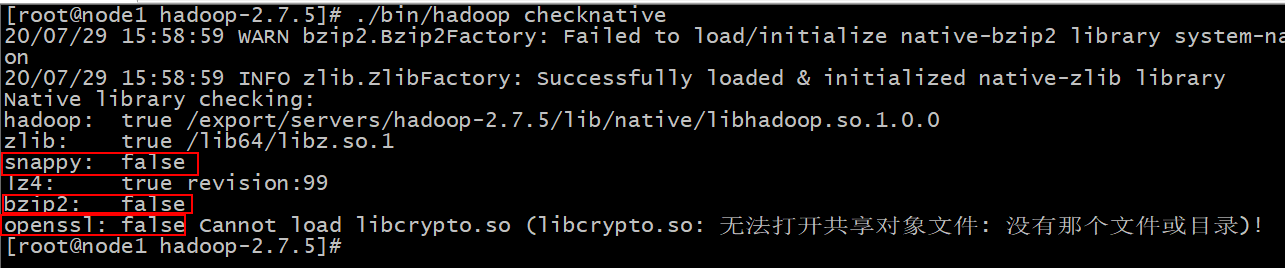
|  |
| --- |
| #!/bin/bash  **tar** **-**zxvf **/**export**/**software**/**jdk-8u241-linux-x64.tar.gz -C **/export/servers/**  **cd** **/export/servers/**jdk1.8.0\_241  home**=`pwd`**  **echo** **$home**  **echo** "export JAVA\_HOME=${home}" **>>** **/**etc**/**profile  **echo** "export PATH=:\$PATH:\$JAVA\_HOME/bin" **>>** **/**etc**/**profile  source **/**etc**/**profile  **for** m **in** 2 3  **do**  scp -r **/export/servers/**jdk1.8.0\_241 node**$m:/export/servers/**  ssh node**$m** "echo 'export JAVA\_HOME=/export/servers/jdk1.8.0\_241' >> /etc/profile; echo 'export PATH=:\$PATH:\$JAVA\_HOME/bin' >> /etc/profile;source /etc/profile"  **done** |

### hahadoop重新编译

#### 为什么要编译hadoop

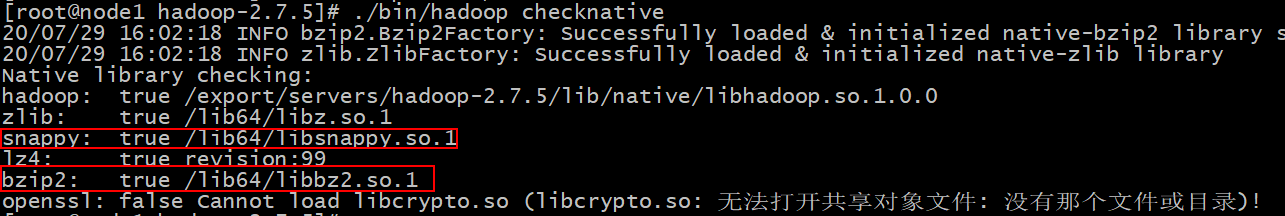
由于appache给出的hadoop的安装包没有提供带C程序访问的接口，所以我们在使用本地库（本地库可以用来做压缩，以及支持C程序等等）的时候就会出问题,需要对Hadoop源码包进行重新编译，请注意，资料中已经提供好了编译过的Hadoop安装包，所以这一部分的操作，大家可以不用做，了解即可。

官方版本中的本地库的截图:



会发现, snappy 和 bzip2 都为false, 表示不支持

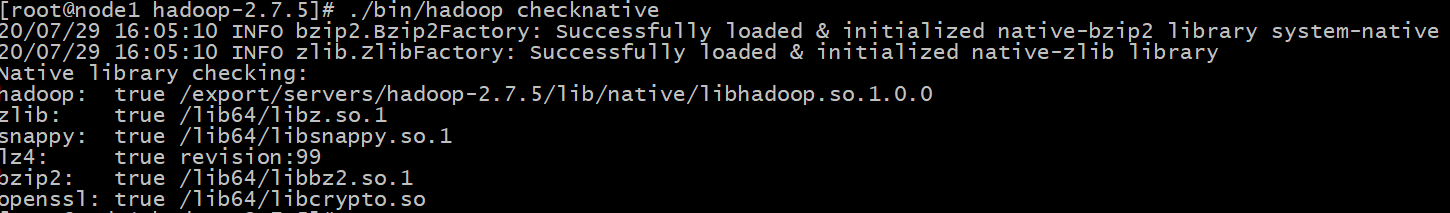
手动编译后的hadoop的版本本地库的截图:



会发现, snappy 和 bzip2 都为false, 表示支持

OpenSSL依然为false, 如何解决呢?

yum -y install openssl-devel



#### Hadoop编译实现

Hadoop的编译步骤可以参考:1\_Hadoop编译文档.docx文档

### Hadoop安装

#### Hadoop安装包目录结构

解压hadoop-2.7.5.tar.gz，目录结构如下：

**bin**：Hadoop最基本的管理脚本和使用脚本的目录，这些脚本是sbin目录下管理脚本的基础实现，用户可以直接使用这些脚本管理和使用Hadoop。

**etc**：Hadoop配置文件所在的目录，包括core-site,xml、hdfs-site.xml、mapred-site.xml等从Hadoop1.0继承而来的配置文件和yarn-site.xml等Hadoop2.0新增的配置文件。

**include**：对外提供的编程库头文件（具体动态库和静态库在lib目录中），这些头文件均是用C++定义的，通常用于C++程序访问HDFS或者编写MapReduce程序。

**lib**：该目录包含了Hadoop对外提供的编程动态库和静态库，与include目录中的头文件结合使用。

**libexec**：各个服务对用的shell配置文件所在的目录，可用于配置日志输出、启动参数（比如JVM参数）等基本信息。

**sbin**：Hadoop管理脚本所在的目录，主要包含HDFS和YARN中各类服务的启动/关闭脚本。

**share**：Hadoop各个模块编译后的jar包所在的目录，官方自带示例。

#### Hadoop配置文件修改

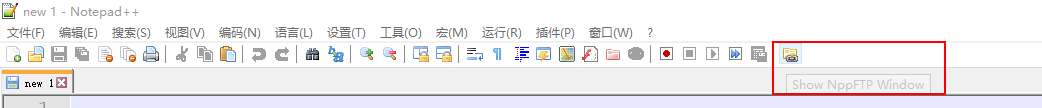
Hadoop安装主要就是配置文件的修改，一般在主节点进行修改，完毕后scp下发给其他各个从节点机器。

注意,以下所有操作都在node1主机进行。

注意: 在进行下面的操作, 请先将编译后的hadoop包上传到linux, 并且解压到/export/server

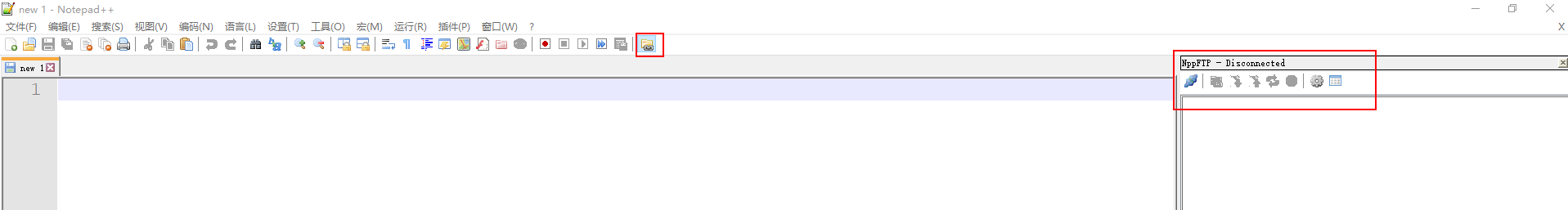
由于要修改的配置文件比较的多, 建议采用notepad++ 连接远程的主机, 在notepad++上直接进行修改保存即可: 如何使用呢

* 1) 观察在notepad++ 上 有没有一下这个图标

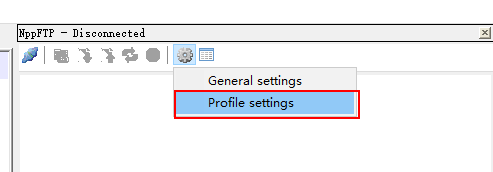


如果没有, 请安装资料中提供的notepad++ 即可

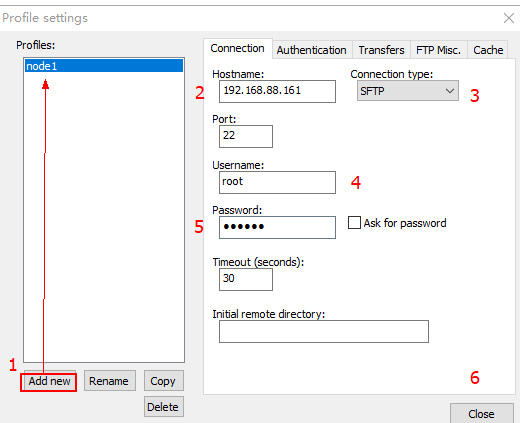
* 2) 点击这个黄色的按钮:



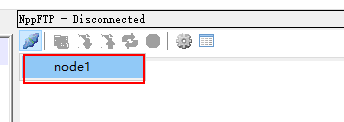
* 3) 点击 设置按钮--profile



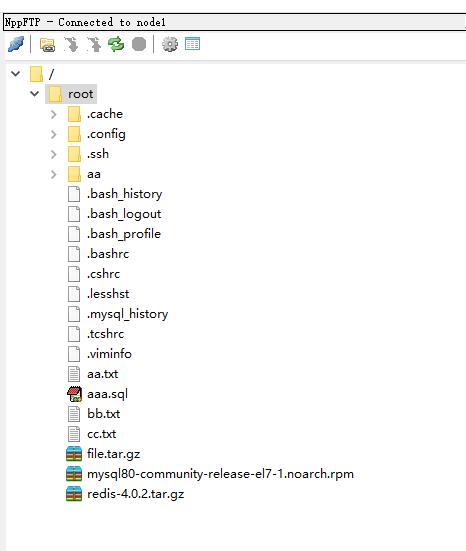
* 4) 开始设置虚拟机的地址即可: 可以将所有的服务器都配置, 也可以只配置第一个



* 5) 点的 connect. 选择要连接的主机即可



* 6) 此时如果弹出一个框框, 直接一顿选择yes 就好了, 完成后, 如果能够看到一下内容, 说明连接成功了



此时就可以愉快的选择对应文件, 进行修改保存操作即可

##### hadoop-env.sh

**1、介绍**

文件中设置的是Hadoop运行时需要的环境变量。JAVA\_HOME是必须设置的，即使我们当前的系统中设置了JAVA\_HOME，它也是不认识的，因为Hadoop即使是在本机上执行，它也是把当前的执行环境当成远程服务器。

**2、配置**

|  |
| --- |
| **cd** **/export/servers/**hadoop-2.7.5**/**etc**/**hadoop  **vim** hadoop-env.sh |

添加以下内容:

|  |
| --- |
| **export** JAVA\_HOME**=/export/servers/**jdk1.8.0\_241 |

##### core-site.xml

**1、介绍**

hadoop的核心配置文件，有默认的配置项core-default.xml。

core-default.xml与core-site.xml的功能是一样的，如果在core-site.xml里没有配置的属性，则会自动会获取core-default.xml里的相同属性的值。

1. **配置**

在该文件中的<configuration>标签中添加以下配置,

<configuration>

在这里添加配置

</configuration>

|  |
| --- |
| **cd** **/export/servers/**hadoop-2.7.5**/**etc**/**hadoop  **vim** core-site.xml |

**配置内容如下:**

|  |
| --- |
| <!-- 用于设置Hadoop的文件系统，由URI指定 -->  <property>  <name>**fs.defaultFS**</name>  <value>**hdfs://node1:8020**</value>  </property>  <!-- 配置Hadoop存储数据目录,默认/tmp/hadoop-${user.name} -->  <property>  <name>**hadoop.tmp.dir**</name>  <value>**/export/servers/hadoop-2.7.5/hadoopDatas/tempDatas**</value>  </property>  <!-- 缓冲区大小，实际工作中根据服务器性能动态调整: 根据自己的虚拟机的内存大小进行配置即可, 不要小于1GB, 最高配置为 4gb -->  <property>  <name>**io.file.buffer.size**</name>  <value>**4096**</value>  </property>  <!-- 开启hdfs的垃圾桶机制，删除掉的数据可以从垃圾桶中回收，单位分钟 -->  <property>  <name>**fs.trash.interval**</name>  <value>**10080**</value>  </property> |

##### hdfs-site.xml

**1、介绍**

HDFS的核心配置文件，主要配置HDFS相关参数，有默认的配置项hdfs-default.xml。

hdfs-default.xml与hdfs-site.xml的功能是一样的，如果在hdfs-site.xml里没有配置的属性，则会自动会获取hdfs-default.xml里的相同属性的值。

**2、配置**

在该文件中的<configuration>标签中添加以下配置,

<configuration>

在这里添加配置

</configuration>

|  |
| --- |
| **cd** **/export/servers/**hadoop-2.7.5**/**etc**/**hadoop  **vim** hdfs-site.xml |

配置一下内容

|  |
| --- |
| <!-- 指定SecondaryNameNode的主机和端口 -->  <property>  <name>dfs.namenode.secondary.http-address</name>  <value>node2:50090</value>  </property>  <!-- 指定namenode的页面访问地址和端口 -->  <property>  <name>dfs.namenode.http-address</name>  <value>node1:50070</value>  </property>  <!-- 指定namenode元数据的存放位置 -->  <property>  <name>dfs.namenode.name.dir</name>  <value>file:///export/servers/hadoop-2.7.5/hadoopDatas/namenodeDatas</value>  </property>  <!-- 定义datanode数据存储的节点位置 -->  <property>  <name>dfs.datanode.data.dir</name>  <value>file:///export/servers/hadoop-2.7.5/hadoopDatas/datanodeDatas</value>  </property>  <!-- 定义namenode的edits文件存放路径 -->  <property>  <name>dfs.namenode.edits.dir</name>  <value>file:///export/servers/hadoop-2.7.5/hadoopDatas/nn/edits</value>  </property>  <!-- 配置检查点目录 -->  <property>  <name>dfs.namenode.checkpoint.dir</name>  <value>file:///export/servers/hadoop-2.7.5/hadoopDatas/snn/name</value>  </property>  <property>  <name>dfs.namenode.checkpoint.edits.dir</name>  <value>file:///export/servers/hadoop-2.7.5/hadoopDatas/dfs/snn/edits</value>  </property>  <!-- 文件切片的副本个数-->  <property>  <name>dfs.replication</name>  <value>3</value>  </property>  <!-- 设置HDFS的文件权限-->  <property>  <name>dfs.permissions</name>  <value>false</value>  </property>  <!-- 设置一个文件切片的大小：128M-->  <property>  <name>dfs.blocksize</name>  <value>134217728</value>  </property>  <!-- 指定DataNode的节点配置文件 -->  <property>  <name>dfs.hosts</name>  <value>/export/servers/hadoop-2.7.5/etc/hadoop/slaves</value>  </property> |

##### mapred-site.xml

**1、介绍**

MapReduce的核心配置文件，Hadoop默认只有个模板文件mapred-site.xml.template,需要使用该文件复制出来一份mapred-site.xml文件

**2、配置**

|  |
| --- |
| **cd** **/export/servers/**hadoop-2.7.5**/**etc**/**hadoop  **cp** mapred-site.xml.template mapred-site.xml |

在mapred-site.xml文件中的<configuration>标签中添加以下配置,

<configuration>

在这里添加配置

</configuration>

|  |
| --- |
| **vim** mapred-site.xml |

配置一下内容:

|  |
| --- |
| <!-- 指定分布式计算使用的框架是yarn -->  <property>  <name>**mapreduce.framework.name**</name>  <value>**yarn**</value>  </property>  <!-- 开启MapReduce小任务模式 -->  <property>  <name>**mapreduce.job.ubertask.enable**</name>  <value>**true**</value>  </property>    <!-- 设置历史任务的主机和端口 -->  <property>  <name>**mapreduce.jobhistory.address**</name>  <value>**node1:10020**</value>  </property>  <!-- 设置网页访问历史任务的主机和端口 -->  <property>  <name>**mapreduce.jobhistory.webapp.address**</name>  <value>**node1:19888**</value>  </property> |

##### mapred-env.sh

在该文件中需要指定JAVA\_HOME,将原文件的JAVA\_HOME配置前边的注释去掉，然后按照以下方式修改:

|  |
| --- |
| **cd** **/export/servers/**hadoop-2.7.5**/**etc**/**hadoop  **vim** mapred-env.sh |

|  |
| --- |
| **export** JAVA\_HOME**=/export/servers/**jdk1.8.0\_241 |

##### yarn-site.xml

YARN的核心配置文件,在该文件中的<configuration>标签中添加以下配置,

<configuration>

在这里添加配置

</configuration>

|  |
| --- |
| **cd** **/export/servers/**hadoop-2.7.5**/**etc**/**hadoop  **vim** yarn-site.xml |

添加以下配置：

|  |
| --- |
| <!-- 配置yarn主节点的位置 -->  <property>  <name>yarn.resourcemanager.hostname</name>  <value>node1</value>  </property>  <property>  <name>yarn.nodemanager.aux-services</name>  <value>mapreduce\_shuffle</value>  </property>    <!-- 开启日志聚合功能 -->  <property>  <name>yarn.log-aggregation-enable</name>  <value>true</value>  </property>  <!-- 设置聚合日志在hdfs上的保存时间 -->  <property>  <name>yarn.log-aggregation.retain-seconds</name>  <value>604800</value>  </property>  <!-- 设置yarn集群的内存分配方案 -->  <property>  <name>yarn.nodemanager.resource.memory-mb</name>  <value>20480</value>  </property>  <property>  <name>yarn.scheduler.minimum-allocation-mb</name>  <value>2048</value>  </property>  <property>  <name>yarn.nodemanager.vmem-pmem-ratio</name>  <value>2.1</value>  </property> |

##### slaves

**1、介绍**

slaves文件里面记录的是集群主机名。一般有以下两种作用：

一是：配合一键启动脚本如start-dfs.sh、stop-yarn.sh用来进行集群启动。这时候slaves文件里面的主机标记的就是从节点角色所在的机器。

二是：可以配合hdfs-site.xml里面dfs.hosts属性形成一种白名单机制。

dfs.hosts指定一个文件，其中包含允许连接到NameNode的主机列表。必须指定文件的完整路径名,那么所有在slaves中的主机才可以加入的集群中。如果值为空，则允许所有主机。

1. **配置**

|  |
| --- |
| **cd** **/export/servers/**hadoop-2.7.5**/**etc**/**hadoop  **vim** slaves |

删除slaves中的localhost，然后添加以下内容:

|  |
| --- |
| node1  node2  node3 |

#### 数据目录创建和文件分发

注意,以下所有操作都在node1主机进行。

**1、目录创建**

创建Hadoop所需目录

|  |
| --- |
| **mkdir** -p **/export/servers/**hadoop-2.7.5**/**hadoopDatas**/**tempDatas  **mkdir** -p **/export/servers/**hadoop-2.7.5**/**hadoopDatas**/**namenodeDatas  **mkdir** -p **/export/servers/**hadoop-2.7.5**/**hadoopDatas**/**datanodeDatas  **mkdir** -p **/export/servers/**hadoop-2.7.5**/**hadoopDatas**/**nn**/**edits  **mkdir** -p **/export/servers/**hadoop-2.7.5**/**hadoopDatas**/**snn**/**name  **mkdir** -p **/export/servers/**hadoop-2.7.5**/**hadoopDatas**/**dfs**/**snn**/**edits |

1. **文件分发**

**将配置好的Hadoop目录分发到node2和node3主机。**

|  |
| --- |
| scp -r /export/servers/hadoop-2.7.5/ node2:/export/servers/  scp -r /export/servers/hadoop-2.7.5/ node3:/export/servers/ |

#### ****配置Hadoop的环境变量****

注意，三台机器都需要执行以下命令

|  |
| --- |
| **vim** **/**etc**/**profile |

添加以下内容:

|  |
| --- |
| **export** HADOOP\_HOME**=/export/servers/**hadoop-2.7.5  **export** PATH**=:$HADOOP\_HOME/**bin**:$HADOOP\_HOME/**sbin**:$PATH** |

配置完成之后生效

|  |
| --- |
| source **/**etc**/**profile |

**测试**

|  |
| --- |
| **hadoop version** |

#### ****启动集群****

##### 启动方式

要启动Hadoop集群，需要启动HDFS和YARN两个集群。

注意：**首次启动HDFS时，必须对其进行格式化操作**。本质上是一些清理和准备工作，因为此时的HDFS在物理上还是不存在的。

在node1上执行格式化指令

|  |
| --- |
| hadoop namenode -format |

##### 单节点逐个启动

在node1主机上使用以下命令启动HDFS NameNode：

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

在node1、node2、node3三台主机上，分别使用以下命令启动HDFS DataNode：

|  |
| --- |
| hadoop-daemon.sh **start** datanode |

在node1主机上使用以下命令启动YARN ResourceManager：

|  |
| --- |
| yarn-daemon.sh **start** resourcemanager |

在node1、node2、node3三台主机上使用以下命令启动YARN nodemanager：

|  |
| --- |
| yarn-daemon.sh **start** nodemanager |

以上脚本位于/export/servers/hadoop-2.7.5/sbin目录下。如果想要停止某个节点上某个角色，只需要把命令中的**start**改为**stop**即可。

##### 脚本一键启动

启动HDFS

|  |
| --- |
| start-dfs.sh |

启动Yarn

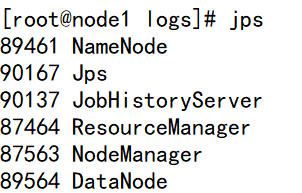
|  |
| --- |
| start-yarn.sh |

启动历史任务服务进程

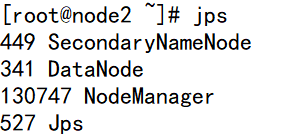
|  |
| --- |
| mr-jobhistory-daemon.sh start historyserver |

启动之后，使用jps命令查看相关服务是否启动，jps是显示Java相关的进程命令。

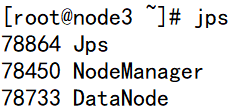
**node1：**



**node2：**

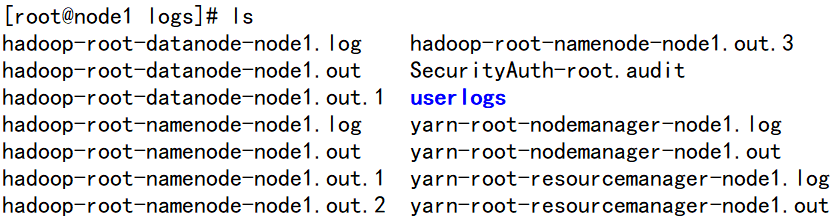


**node3**



停止集群：stop-dfs.sh、stop-yarn.sh、mr-jobhistory-daemon.sh stophistoryservers

注意:如果在启动之后，有些服务没有启动成功，则需要查看启动日志，Hadoop的启动日志在每台主机的/export/servers/hadoop-2.7.5/logs/目录，需要根据哪台主机的哪个服务启动情况去对应的主机上查看相应的日志，以下是node1主机的日志目录.



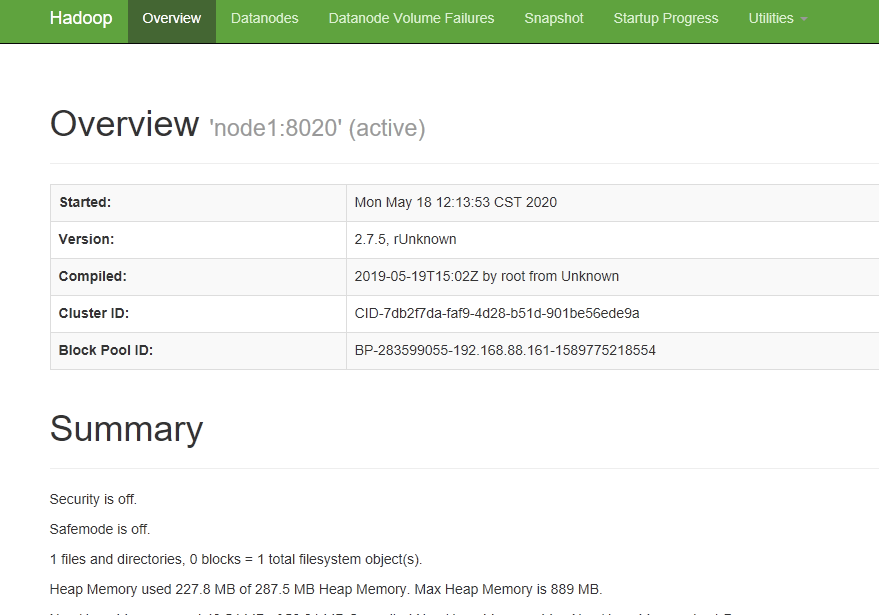
#### 集群的页面访问

##### IP访问

一旦Hadoop集群启动并运行，可以通过web-ui进行集群查看，如下所述：

查看NameNode页面地址:

[**http://192.168.88.161:50070/**](#/)



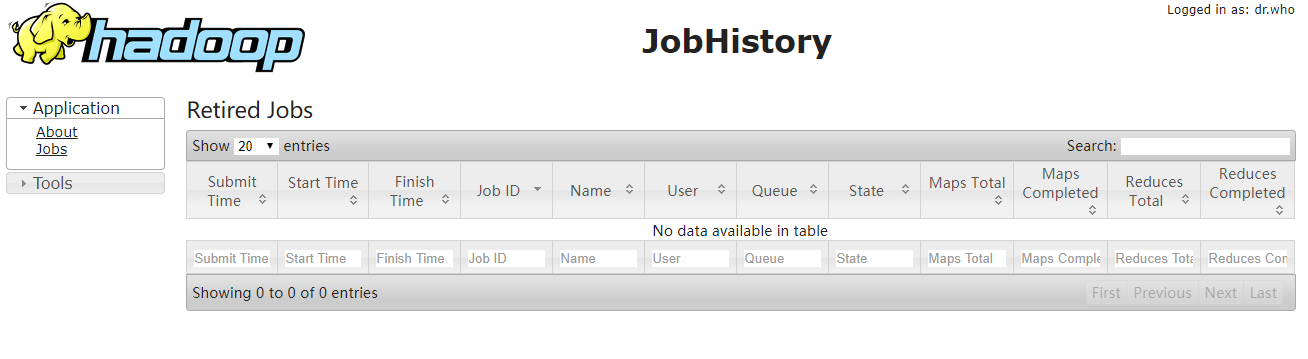
查看Yarn集群页面地址:

[**http://192.168.88.161:8088/cluster**](http://node01:8088/cluster)



查看MapReduce历史任务页面地址:

[**http://192.168.88.161:19888/jobhistory**](http://node01:19888/jobhistory)



##### 主机名访问

请注意，以上的访问地址只能使用IP地址，如果想要使用主机名，则对Windows进行配置。

配置方式:

1、打开Windows的C:\Windows\System32\drivers\etc目录下hosts文件

2、在hosts文件中添加以下域名映射

|  |
| --- |
| 192.168.88.161 node1 node1**.**itcast**.**cn  192.168.88.162 node2 node2**.**itcast**.**cn  192.168.88.163 node3 node3**.**itcast**.**cn |

配置完之后，可以将以上地址中的IP替换为主机名即可访问，如果还不能访问，则需要重启Windows电脑，比如访问NameNode，可以使用**[http://node1:50070/](" \l "/)** 。

#### Hadoop初体验

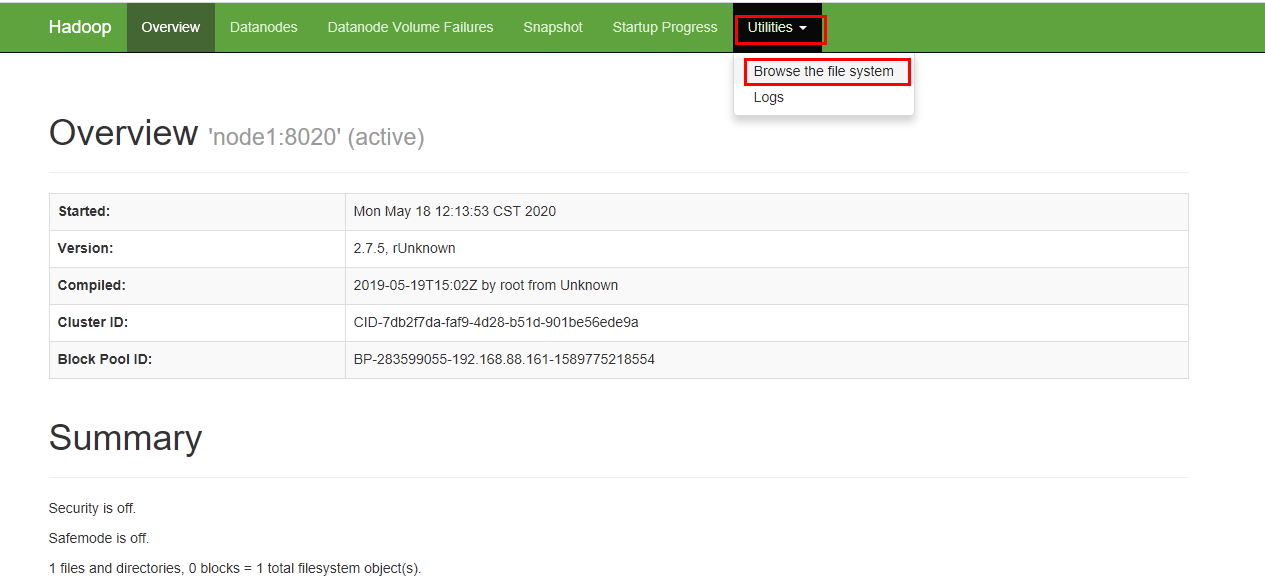
##### HDFS使用

1、从Linux本地上传一个文本文件到hdfs的/目录下

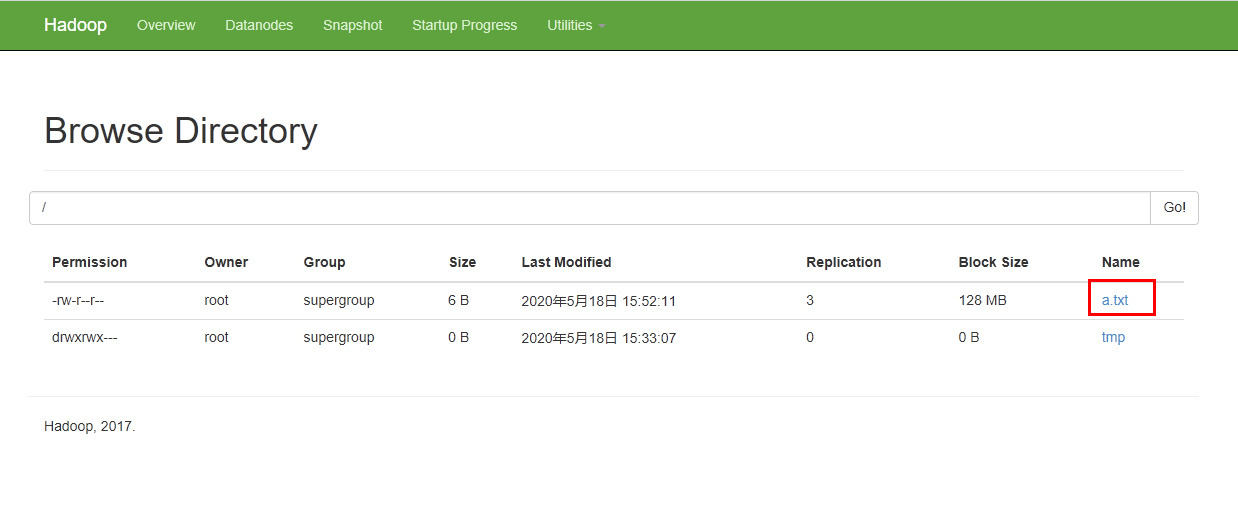
|  |
| --- |
| #在/export/data/目录中创建a.txt文件，并写入数据  **cd** **/**export**/**data**/**  **touch** a.txt  **echo** "hello" **>** a.txt  #将a.txt上传到HDFS的根目录  hadoop fs **-**put a.txt **/** |

1. 通过页面查看

通过NameNode页面.进入HDFS：**[http://node1:50070/](" \l "/)**



查看文件是否创建成功.



##### 运行mapreduce程序

在Hadoop安装包的share/hadoop/mapreduce下有官方自带的mapreduce程序。我们可以使用如下的命令进行运行测试。

示例程序jar:

hadoop-mapreduce-examples-2.7.5.jar

计算圆周率

|  |
| --- |
| hadoop jar **/export/servers/**hadoop-2.7.5**/**share**/**hadoop**/**mapreduce**/**hadoop-mapreduce-examples-2.7.5.jar pi 2 50 |

关于圆周率的估算，感兴趣的可以查询资料蒙特卡洛方法来计算Pi值，计算命令中2表示计算的线程数，50表示投点数，该值越大，则计算的pi值越准确。

