目录

[课程大纲（HADOOP快速入门） 2](#_Toc439057207)

[1. HADOOP 快速入门 3](#_Toc439057208)

[什么是HADOOP 3](#_Toc439057209)

[HADOOP产生背景 3](#_Toc439057210)

[HADOOP在大数据、云计算中的位置和关系 3](#_Toc439057211)

[国内外HADOOP应用案例介绍 4](#_Toc439057212)

[国内HADOOP的就业情况分析 5](#_Toc439057213)

[HADOOP生态圈以及各组成部分的简介 6](#_Toc439057214)

[分布式系统概述 6](#_Toc439057215)

[2. HIVE快速入门 7](#_Toc439057216)

[2.1 Hive基本介绍 7](#_Toc439057217)

[2.2 Hive的基本使用 8](#_Toc439057218)

[2.3 数据仓库基本知识 9](#_Toc439057219)

[3. 数据分析案列演示 10](#_Toc439057220)

[3.1 需求分析 10](#_Toc439057221)

[3.1.1案例名称 10](#_Toc439057222)

[3.1.2 案例需求描述 10](#_Toc439057223)

[3.1.3 web点击流日志的数据格式 10](#_Toc439057224)

[3.1.4 分析指标 11](#_Toc439057225)

[3.1.5 统计结果数据可视化 11](#_Toc439057226)

[3.2 数据来源分析 12](#_Toc439057227)

[3.2.1 企业中获取数据的几种方式 12](#_Toc439057228)

[3.2.2 数据采集 12](#_Toc439057229)

[3.3 数据处理流程 13](#_Toc439057230)

[数据预处理/加载入库 13](#_Toc439057231)

[使用Hive做数据ETL 14](#_Toc439057232)

[使用Hive运算业务指标 16](#_Toc439057233)

[将结果数据导出到mysql（sqoop） 17](#_Toc439057234)

[结果展现——数据可视化 17](#_Toc439057235)

[4. 集群搭建 18](#_Toc439057236)

[4.1 HADOOP集群搭建 18](#_Toc439057237)

[4.1.1集群简介： 18](#_Toc439057238)

[4.1.2服务器准备 18](#_Toc439057239)

[4.1.3网络环境准备 18](#_Toc439057240)

[4.1.4服务器系统设置 18](#_Toc439057241)

[4.1.5 Jdk环境安装 19](#_Toc439057242)

[4.1.6 HADOOP安装部署 19](#_Toc439057243)

[4.1.7 启动集群 21](#_Toc439057244)

[4.1.8 测试 21](#_Toc439057245)

[4.2 Hive搭建 22](#_Toc439057246)

[Hive的配置安装 22](#_Toc439057247)

[Hive的使用 23](#_Toc439057248)

[Hive运行测试 23](#_Toc439057249)

# 课程大纲（HADOOP快速入门）

|  |  |  |
| --- | --- | --- |
| HADOOP快速入门 | HADOOP快速入门 | HADOOP产生背景 |
| HADOOP在大数据、云计算中的位置和关系 |
| 国内外HADOOP应用案例介绍 |
| 国内HADOOP的就业情况分析及课程大纲介绍 |
| 分布式系统概述 |
| HADOOP生态圈以及各组成部分的简介 |
| Hive快速入门 | Hive快速入门 | Hive基本介绍 |
| Hive的使用 |
| 数据仓库基本知识 |
| 数据分析案例演示 | 需求分析 | 案列：定义需求、介绍数据格式 |
| 数据获取 | 企业中获取数据的几种方式 |
| 将文件直接导入到数据仓库 |
| 将数据库的数据导入到数据仓库（sqoop） |
| 数据处理 | 使用Hive对数进行清洗（ETL的过程） |
| 数据计算 | 使用Hive对数据进行计算 |
| 数据展现 | 将结果数据导出到mysql（sqoop） |

学习目标：

第一天接触具体的大数据框架，总目标是让学习者建立起大数据和分布式的宏观概念

1、理解hadoop是什么，用于做什么，大体上怎么用

2、理解hive是什么，用于做什么，大体上怎么用

3、通过一个案例的演示说明，理解数据挖掘系统的基本流程和结构

# 1. HADOOP背景介绍

## 1.1 什么是HADOOP

1. HADOOP是apache旗下的一套开源**软件平台**
2. HADOOP提供的功能：利用**服务器集群**，根据用户的自定义业务逻辑，**对海量数据进行分布式处理**
3. HADOOP的核心组件有
   * 1. HDFS（分布式文件系统）
     2. YARN（运算资源调度系统）
     3. MAPREDUCE（分布式运算编程框架）
4. 广义上来说，HADOOP通常是指一个更广泛的概念——HADOOP生态圈

## 1.2 HADOOP产生背景

1. HADOOP**最早起源于Nutch**。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。
2. 2003年、2004年**谷歌发表的两篇论文为该问题提供了可行的解决方案**。

——分布式文件系统（GFS），可用于处理海量网页的**存储**

——分布式计算框架MAPREDUCE，可用于处理海量网页的**索引计算**问题。

1. 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。

## 1.3 HADOOP在大数据、云计算中的位置和关系

1. **云计算**是分布式计算、并行计算、网格计算、多核计算、网络存储、虚拟化、负载均衡等传统计算机技术和互联网技术融合发展的产物。借助**IaaS(基础设施即服务)、PaaS(平台即服务)、SaaS（软件即服务）**等业务模式，把强大的计算能力提供给终端用户。
2. 现阶段，云计算的**两大底层支撑技术**为“**虚拟化**”和“**大数据技术**”
3. 而HADOOP则是云计算的PaaS层的解决方案之一，并不等同于PaaS，更不等同于云计算本身。

## 1.4 国内外HADOOP应用案例介绍

**1、HADOOP应用于数据服务基础平台建设**



**2/HADOOP用于用户画像**



**3、HADOOP用于网站点击流日志数据挖掘**



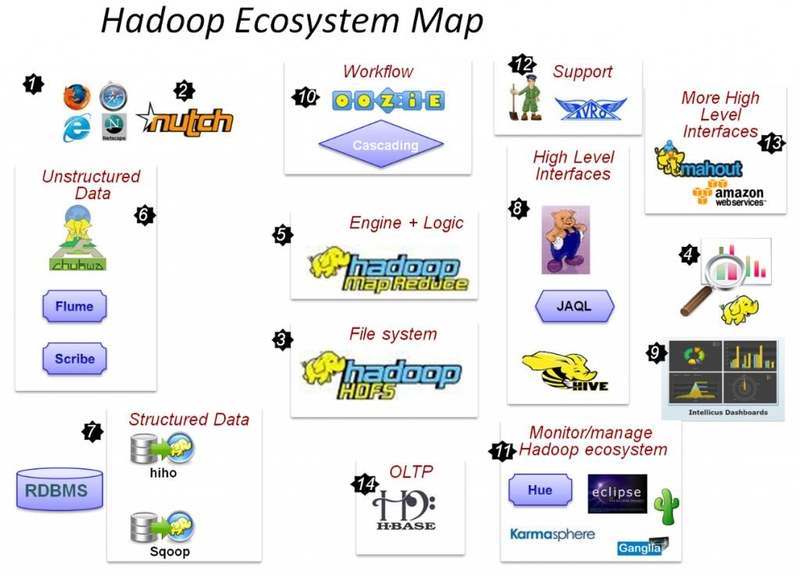
## 1.5 国内HADOOP的就业情况分析

1. HADOOP就业整体情况
2. 大数据产业已纳入**国家十三五规划**
3. 各大城市都在进行**智慧城市项目**建设，而智慧城市的根基就是大数据综合平台
4. 互联网时代数据的种类，增长都呈现**爆发式增长**，各行业对数据的价值日益重视
5. 相对于传统JAVAEE技术领域来说，大数据领域的**人才相对稀缺**
6. 随着现代社会的发展，数据处理和数据挖掘的重要性只会增不会减，因此，大数据技术是一个尚在蓬勃发展且具有**长远前景的领域**
7. HADOOP就业职位要求

大数据是个复合专业，包括应用开发、软件平台、算法、数据挖掘等，因此，**大数据技术领域的就业选择是多样的**，但就HADOOP而言，通常都需要具备以下技能或知识：

1. HADOOP分布式集群的平台搭建
2. HADOOP分布式文件系统HDFS的原理理解及使用
3. HADOOP分布式运算框架MAPREDUCE的原理理解及编程
4. Hive数据仓库工具的熟练应用
5. Flume、sqoop、oozie等辅助工具的熟练使用
6. Shell/python等脚本语言的开发能力

## 1.6 HADOOP生态圈以及各组成部分的简介



各组件简介

重点组件：

HDFS：分布式文件系统

MAPREDUCE：分布式运算程序开发框架

HIVE：基于大数据技术（文件系统+运算框架）的SQL数据仓库工具

HBASE：基于HADOOP的分布式海量数据库

**ZOOKEEPER：分布式协调服务基础组件**

Mahout：基于mapreduce/spark/flink等分布式运算框架的机器学习算法库

Oozie：工作流调度框架

Sqoop：数据导入导出工具

Flume：日志数据采集框架

# 2 分布式系统概述

*注：由于大数据技术领域的各类技术框架基本上都是分布式系统，因此，理解hadoop、storm、spark等技术框架，都需要具备基本的分布式系统概念*

## 2.1 分布式软件系统(Distributed Software Systems)

* 该软件系统会划分成多个子系统或模块，各自运行在不同的机器上，子系统或模块之间通过网络通信进行协作，实现最终的整体功能
* 比如分布式操作系统、分布式程序设计语言及其编译(解释)系统、分布式文件系统和分布式数据库系统等。

## 2.2 分布式软件系统举例：solrcloud

1. 一个solrcloud集群通常有多台solr服务器
2. 每一个solr服务器节点负责存储整个索引库的若干个shard（数据分片）
3. 每一个shard又有多台服务器存放若干个副本互为主备用
4. 索引的建立和查询会在整个集群的各个节点上并发执行
5. solrcloud集群作为整体对外服务，而其内部细节可对客户端透明

**总结：利用多个节点共同协作完成一项或多项具体业务功能的系统就是分布式系统。**

## 2.3 分布式应用系统模拟开发

**需求：**可以实现由主节点将运算任务发往从节点，并将各从节点上的任务启动；

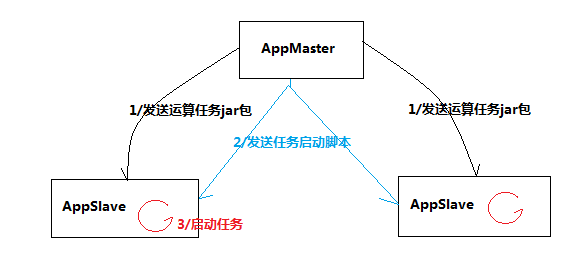
**程序清单：**

AppMaster

AppSlave/APPSlaveThread

Task

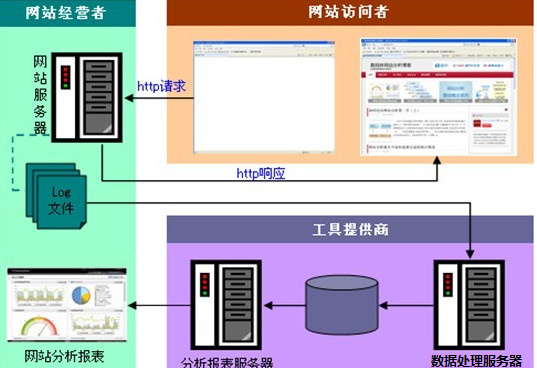
**程序运行逻辑流程：**



# 3. 离线数据分析流程介绍

*注：本环节主要感受数据分析系统的宏观概念及处理流程，初步理解hadoop等框架在其中的应用环节，不用过于关注代码细节*

一个应用广泛的数据分析系统：“web日志数据挖掘”



## 3.1 需求分析

### 3.1.1 案例名称

“网站或APP点击流日志数据挖掘系统”。

### 3.1.2 案例需求描述

“Web点击流日志”包含着网站运营很重要的信息，通过日志分析，我们可以知道网站的访问量，哪个网页访问人数最多，哪个网页最有价值，广告转化率、访客的来源信息，访客的终端信息等。

### 3.1.3 数据来源

本案例的数据主要由**用户的点击行为记录**

获取方式：在页面预埋一段js程序，为页面上想要监听的标签绑定事件，只要用户点击或移动到标签，即**可触发ajax请求到后台servlet程序**，**用log4j记录下事件信息**，从而在web服务器（nginx、tomcat等）上形成不断增长的日志文件。

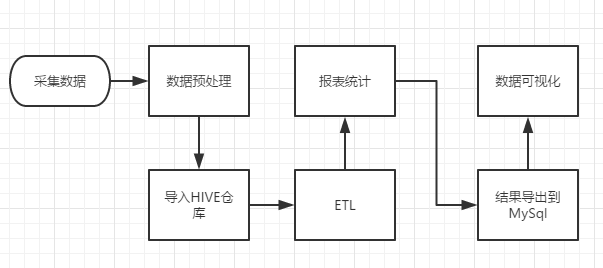
形如：

|  |
| --- |
| 58.215.204.118 - - [18/Sep/2013:06:51:35 +0000] "GET /wp-includes/js/jquery/jquery.js?ver=1.10.2 HTTP/1.1" 304 0 "http://blog.fens.me/nodejs-socketio-chat/" "Mozilla/5.0 (Windows NT 5.1; rv:23.0) Gecko/20100101 Firefox/23.0" |

## 3.2 数据处理流程

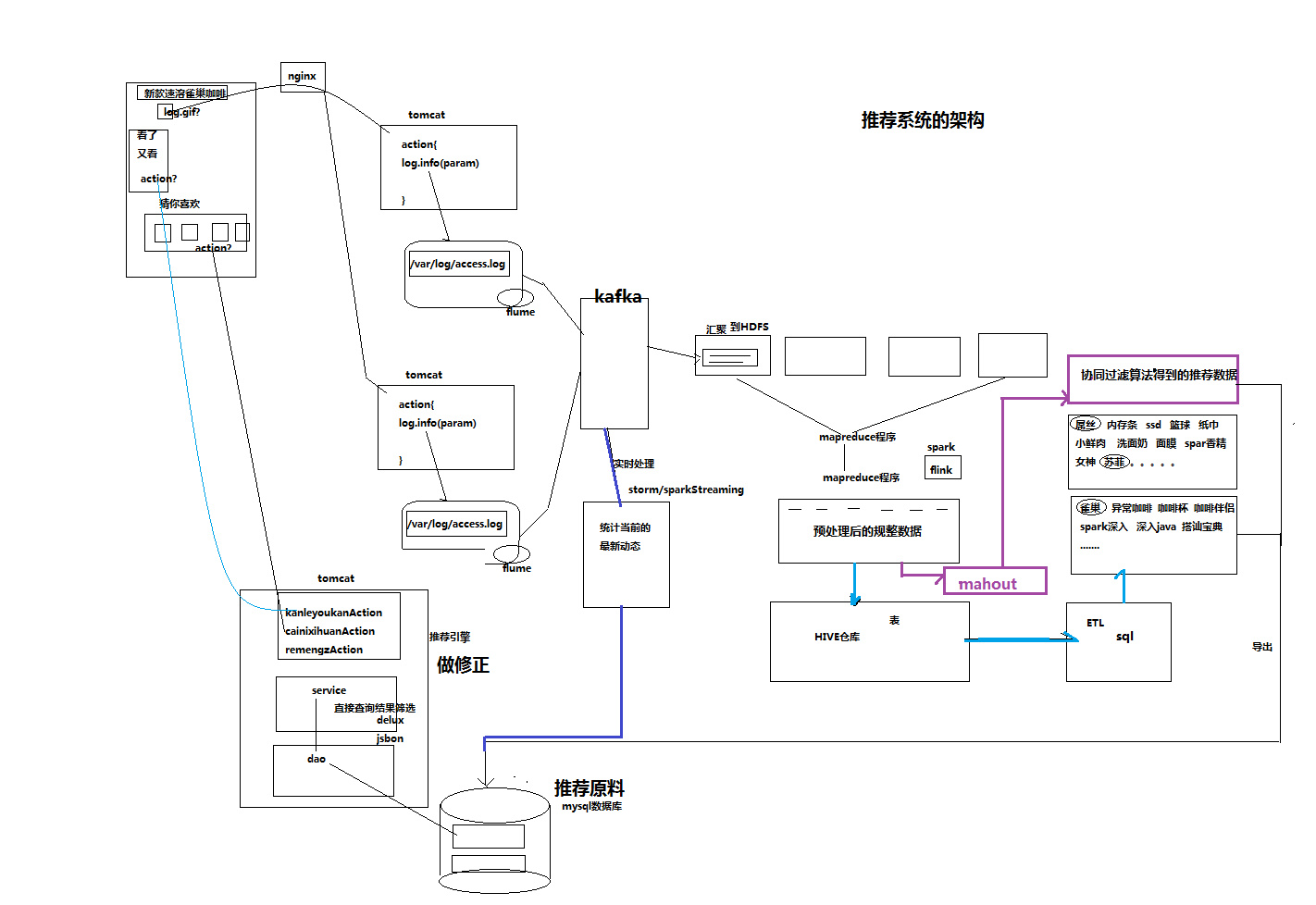
### 3.2.1 流程图解析

本案例跟典型的BI系统极其类似，整体流程如下：

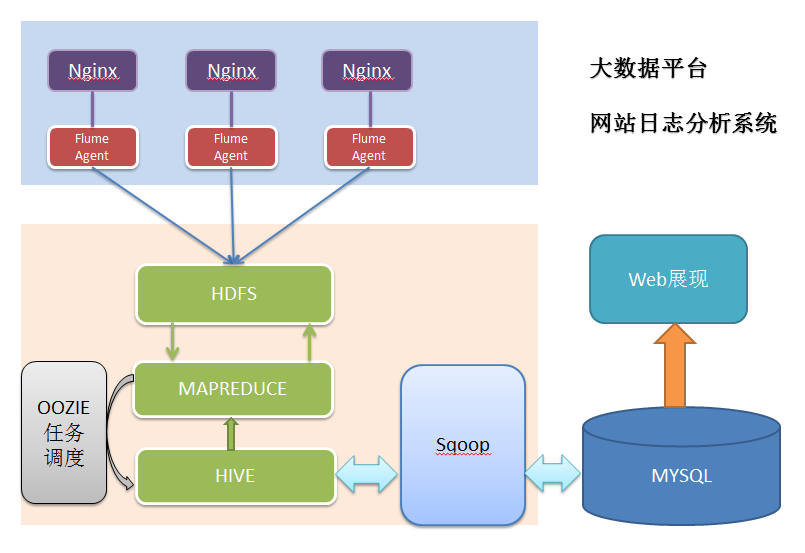


但是，由于本案例的前提是处理海量数据，因而，流程中各环节所使用的技术则跟传统BI完全不同，后续课程都会一一讲解：

1. 数据采集：定制开发**采集程序**，或**使用开源框架FLUME**
2. 数据预处理：定制开发mapreduce程序运行于hadoop集群
3. 数据仓库技术：基于hadoop之上的Hive
4. 数据导出：基于hadoop的sqoop数据导入导出工具
5. 数据可视化：定制开发web程序或使用kettle等产品
6. 整个过程的流程调度：hadoop生态圈中的oozie工具或其他类似开源产品



### 3.2.2 项目技术架构图



# 4. 集群搭建

## 4.1 HADOOP集群搭建

### 4.1.1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**

**YARN集群：**

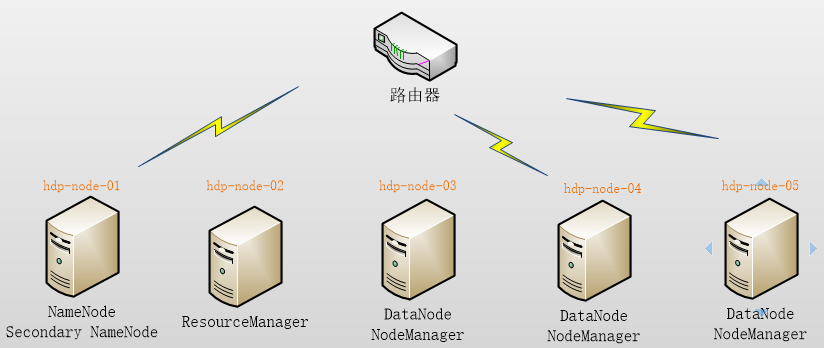
**负责海量数据运算时的资源调度，集群中的角色主要有 ResourceManager /NodeManager**

*(那mapreduce是什么呢？它其实是一个应用程序开发包)*

本集群搭建案例，以5节点为例进行搭建，角色分配如下：

|  |
| --- |
| hdp-node-01 NameNode SecondaryNameNode  hdp-node-02 ResourceManager  hdp-node-03 DataNode NodeManager  hdp-node-04 DataNode NodeManager  hdp-node-05 DataNode NodeManager |

部署图如下：



### 4.1.2服务器准备

本案例使用虚拟机服务器来搭建HADOOP集群，所用软件及版本：

* Vmware 11.0
* Centos 6.5 64bit

### 4.1.3网络环境准备

* 采用NAT方式联网
* 网关地址：192.168.33.1
* 3个服务器节点IP地址：192.168.33.101、192.168.33.102、192.168.33.103
* 子网掩码：255.255.255.0

### 4.1.4服务器系统设置

* 添加HADOOP用户
* 为HADOOP用户分配sudoer权限
* 同步时间
* 设置主机名
  + hdp-node-01
  + hdp-node-02
  + hdp-node-03
* 配置内网域名映射：
  + 192.168.33.101 hdp-node-01
  + 192.168.33.102 hdp-node-02
  + 192.168.33.103 hdp-node-03
* 配置ssh免密登陆
* 配置防火墙

### 4.1.5 Jdk环境安装

* 上传jdk安装包
* 规划安装目录 /home/hadoop/apps/jdk\_1.7.65
* 解压安装包
* 配置环境变量 /etc/profile

### 4.1.6 HADOOP安装部署

* 上传HADOOP安装包
* 规划安装目录 /home/hadoop/apps/hadoop-2.6.1
* 解压安装包
* 修改配置文件 $HADOOP\_HOME/etc/hadoop/

最简化配置如下：

vi hadoop-env.sh

|  |
| --- |
| # The java implementation to use.  export JAVA\_HOME=/home/hadoop/apps/jdk1.7.0\_51 |

vi core-site.xml

|  |
| --- |
| <configuration>  <property>  <name>fs.defaultFS</name>  <value>hdfs://hdp-node-01:9000</value>  </property>  <property>  <name>hadoop.tmp.dir</name>  <value>/home/HADOOP/apps/hadoop-2.6.1/tmp</value>  </property>  </configuration> |

vi hdfs-site.xml

|  |
| --- |
| <configuration>  <property>  <name>dfs. name.dir</name>  <value>/home/hadoop/data/name</value>  </property>  <property>  <name>dfs.data.dir</name>  <value>/home/hadoop/data/data</value>  </property>  <property>  <name>dfs.replication</name>  <value>3</value>  </property>  <property>  <name>dfs.secondary.http.address</name>  <value>hdp-node-01:50090</value>  </property>  </configuration> |

vi mapred-site.xml

|  |
| --- |
| <configuration>  <property>  <name>mapreduce.framework.name</name>  <value>yarn</value>  </property>  </configuration> |

vi yarn-site.xml

|  |
| --- |
| <configuration>  <property>  <name>yarn.resourcemanager.hostname</name>  <value>hadoop01</value>  </property>  <property>  <name>yarn.nodemanager.aux-services</name>  <value>mapreduce\_shuffle</value>  </property>  </configuration> |

vi salves

|  |
| --- |
| hdp-node-01  hdp-node-02  hdp-node-03 |

### 4.1.7 启动集群

初始化HDFS

|  |
| --- |
| bin/hadoop namenode -format |

启动HDFS

|  |
| --- |
| sbin/start-dfs.sh |

启动YARN

|  |
| --- |
| sbin/start-yarn.sh |

### 4.1.8 测试

#### 1、上传文件到HDFS

从本地上传一个文本文件到hdfs的/wordcount/input目录下

|  |
| --- |
| [HADOOP@hdp-node-01 ~]$ HADOOP fs -mkdir -p /wordcount/input  [HADOOP@hdp-node-01 ~]$ HADOOP fs -put /home/HADOOP/somewords.txt /wordcount/input |

#### 2、运行一个mapreduce程序

在HADOOP安装目录下，运行一个示例mr程序

|  |
| --- |
| cd $HADOOP\_HOME/share/hadoop/mapreduce/  hadoop jar mapredcue-example-2.6.1.jar wordcount /wordcount/input /wordcount/output |

1.准备Linux环境

1.0先将虚拟机的网络模式选为NAT

1.1修改主机名

vi /etc/sysconfig/network

NETWORKING=yes

HOSTNAME=itcast ###

1.2修改IP

两种方式：

第一种：通过Linux图形界面进行修改（强烈推荐）

进入Linux图形界面 -> 右键点击右上方的两个小电脑 -> 点击Edit connections -> 选中当前网络System eth0 -> 点击edit按钮 -> 选择IPv4 -> method选择为manual -> 点击add按钮 -> 添加IP：192.168.1.101 子网掩码：255.255.255.0 网关：192.168.1.1 -> apply

第二种：修改配置文件方式（屌丝程序猿专用）

vim /etc/sysconfig/network-scripts/ifcfg-eth0

DEVICE="eth0"

BOOTPROTO="static" ###

HWADDR="00:0C:29:3C:BF:E7"

IPV6INIT="yes"

NM\_CONTROLLED="yes"

ONBOOT="yes"

TYPE="Ethernet"

UUID="ce22eeca-ecde-4536-8cc2-ef0dc36d4a8c"

IPADDR="192.168.1.101" ###

NETMASK="255.255.255.0" ###

GATEWAY="192.168.1.1" ###

1.3修改主机名和IP的映射关系

vim /etc/hosts

192.168.1.101 itcast

1.4关闭防火墙

#查看防火墙状态

service iptables status

#关闭防火墙

service iptables stop

#查看防火墙开机启动状态

chkconfig iptables --list

#关闭防火墙开机启动

chkconfig iptables off

1.5 修改sudo

su root

vim /etc/sudoers

给hadoop用户添加执行的权限

关闭linux服务器的图形界面：

vi /etc/inittab

1.5重启Linux

reboot

2.安装JDK

2.1上传alt+p 后出现sftp窗口，然后put d:\xxx\yy\ll\jdk-7u\_65-i585.tar.gz

2.2解压jdk

#创建文件夹

mkdir /home/hadoop/app

#解压

tar -zxvf jdk-7u55-linux-i586.tar.gz -C /home/hadoop/app

2.3将java添加到环境变量中

vim /etc/profile

#在文件最后添加

export JAVA\_HOME=/home/hadoop/app/jdk-7u\_65-i585

export PATH=$PATH:$JAVA\_HOME/bin

#刷新配置

source /etc/profile

3.安装hadoop2.4.1

先上传hadoop的安装包到服务器上去/home/hadoop/

注意：hadoop2.x的配置文件$HADOOP\_HOME/etc/hadoop

伪分布式需要修改5个配置文件

3.1配置hadoop

第一个：hadoop-env.sh

vim hadoop-env.sh

#第27行

export JAVA\_HOME=/usr/java/jdk1.7.0\_65

第二个：core-site.xml

<!-- 指定HADOOP所使用的文件系统schema（URI），HDFS的老大（NameNode）的地址 -->

<property>

<name>fs.defaultFS</name>

<value>hdfs://weekend-1206-01:9000</value>

</property>

<!-- 指定hadoop运行时产生文件的存储目录 -->

<property>

<name>hadoop.tmp.dir</name>

<value>/home/hadoop/hadoop-2.4.1/tmp</value>

</property>

第三个：hdfs-site.xml

<!-- 指定HDFS副本的数量 -->

<property>

<name>dfs.replication</name>

<value>1</value>

</property>

<property>

<name>dfs.secondary.http.address</name>

<value>192.168.1.152:50090</value>

</property>

第四个：mapred-site.xml (mv mapred-site.xml.template mapred-site.xml)

mv mapred-site.xml.template mapred-site.xml

vim mapred-site.xml

<!-- 指定mr运行在yarn上 -->

<property>

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

<value>yarn</value>

</property>

第五个：yarn-site.xml

<!-- 指定YARN的老大（ResourceManager）的地址 -->

<property>

<name>yarn.resourcemanager.hostname</name>

<value>weekend-1206-01</value>

</property>

<!-- reducer获取数据的方式 -->

<property>

<name>yarn.nodemanager.aux-services</name>

<value>mapreduce\_shuffle</value>

</property>

3.2将hadoop添加到环境变量

vim /etc/proflie

export JAVA\_HOME=/usr/java/jdk1.7.0\_65

export HADOOP\_HOME=/itcast/hadoop-2.4.1

export PATH=$PATH:$JAVA\_HOME/bin:$HADOOP\_HOME/bin:$HADOOP\_HOME/sbin

source /etc/profile

3.3格式化namenode（是对namenode进行初始化）

hdfs namenode -format (hadoop namenode -format)

3.4启动hadoop

先启动HDFS

sbin/start-dfs.sh

再启动YARN

sbin/start-yarn.sh

3.5验证是否启动成功

使用jps命令验证

27408 NameNode

28218 Jps

27643 SecondaryNameNode

28066 NodeManager

27803 ResourceManager

27512 DataNode

http://192.168.1.101:50070 （HDFS管理界面）

http://192.168.1.101:8088 （MR管理界面）

4.配置ssh免登陆

#生成ssh免登陆密钥

#进入到我的home目录

cd ~/.ssh

ssh-keygen -t rsa （四个回车）

执行完这个命令后，会生成两个文件id\_rsa（私钥）、id\_rsa.pub（公钥）

将公钥拷贝到要免密登陆的目标机器上

ssh-copy-id localhost

---------------------------

ssh免登陆：

生成key:

ssh-keygen

复制从A复制到B上:

ssh-copy-id B

验证：

ssh localhost/exit，ps -e|grep ssh

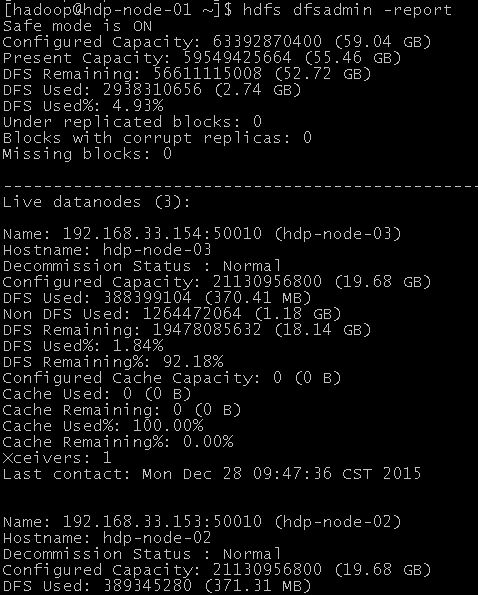
ssh A #在B中执行

# 5 集群使用初步

## 5.1 HDFS使用

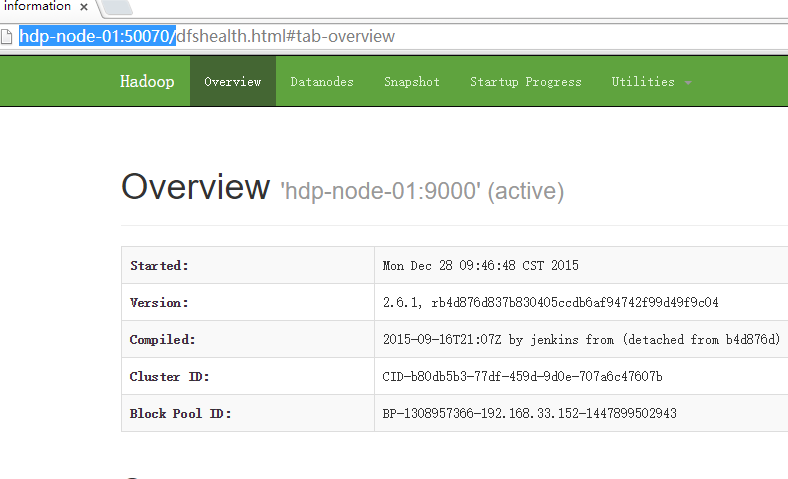
1、查看集群状态

命令： hdfs dfsadmin –report



可以看出，集群共有3个datanode可用

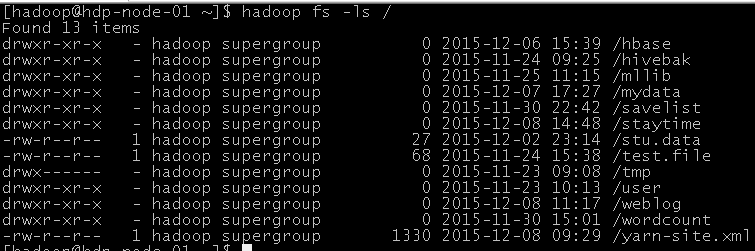
也可打开web控制台查看HDFS集群信息，在浏览器打开<http://hdp-node-01:50070/>



2、上传文件到HDFS

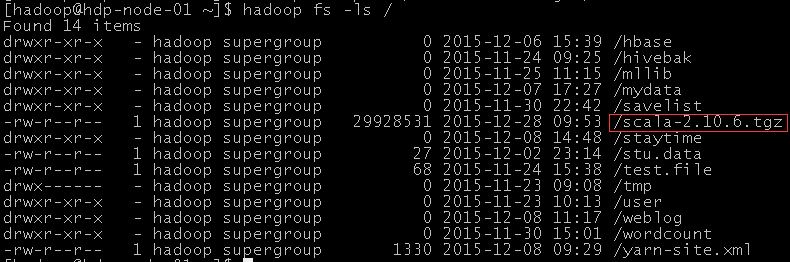
* 查看HDFS中的目录信息

命令： hadoop fs –ls /



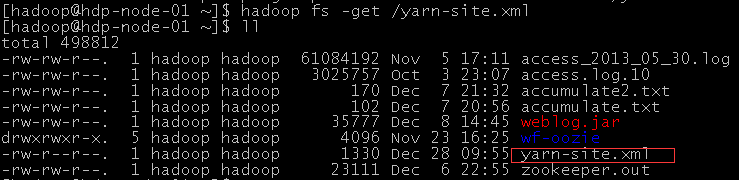
* 上传文件

命令： hadoop fs -put ./ scala-2.10.6.tgz to /



* 从HDFS下载文件

命令： hadoop fs -get /yarn-site.xml



## 5.2 MAPREDUCE使用

mapreduce是hadoop中的分布式运算编程框架，只要按照其编程规范，只需要编写少量的业务逻辑代码即可实现一个强大的海量数据并发处理程序

### 5.2.1 Demo开发——wordcount

1、需求

从大量（比如T级别）文本文件中，统计出每一个单词出现的总次数

2、mapreduce实现思路

Map阶段：

1. 从HDFS的源数据文件中逐行读取数据
2. 将每一行数据切分出单词
3. 为每一个单词构造一个键值对(单词，1)
4. 将键值对发送给reduce

Reduce阶段：

1. 接收map阶段输出的单词键值对
2. 将相同单词的键值对汇聚成一组
3. 对每一组，遍历组中的所有“值”，累加求和，即得到每一个单词的总次数
4. 将(单词，总次数)输出到HDFS的文件中
5. 具体编码实现

(1)定义一个mapper类

|  |
| --- |
| //首先要定义四个泛型的类型  //keyin: LongWritable valuein: Text  //keyout: Text valueout:IntWritable  public class WordCountMapper extends Mapper<LongWritable, Text, Text, IntWritable>{  //map方法的生命周期： 框架每传一行数据就被调用一次  //key : 这一行的起始点在文件中的偏移量  //value: 这一行的内容  @Override  protected void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {  //拿到一行数据转换为string  String line = value.toString();  //将这一行切分出各个单词  String[] words = line.split(" ");  //遍历数组，输出<单词，1>  for(String word:words){  context.write(new Text(word), new IntWritable(1));  }  }  } |

(2)定义一个reducer类

|  |
| --- |
| //生命周期：框架每传递进来一个kv 组，reduce方法被调用一次  @Override  protected void reduce(Text key, Iterable<IntWritable> values, Context context) throws IOException, InterruptedException {  //定义一个计数器  int count = 0;  //遍历这一组kv的所有v，累加到count中  for(IntWritable value:values){  count += value.get();  }  context.write(key, new IntWritable(count));  }  } |

(3)定义一个主类，用来描述job并提交job

|  |
| --- |
| public class WordCountRunner {  //把业务逻辑相关的信息（哪个是mapper，哪个是reducer，要处理的数据在哪里，输出的结果放哪里。。。。。。）描述成一个job对象  //把这个描述好的job提交给集群去运行  public static void main(String[] args) throws Exception {  Configuration conf = new Configuration();  Job wcjob = Job.getInstance(conf);  //指定我这个job所在的jar包  // wcjob.setJar("/home/hadoop/wordcount.jar");  wcjob.setJarByClass(WordCountRunner.class);    wcjob.setMapperClass(WordCountMapper.class);  wcjob.setReducerClass(WordCountReducer.class);  //设置我们的业务逻辑Mapper类的输出key和value的数据类型  wcjob.setMapOutputKeyClass(Text.class);  wcjob.setMapOutputValueClass(IntWritable.class);  //设置我们的业务逻辑Reducer类的输出key和value的数据类型  wcjob.setOutputKeyClass(Text.class);  wcjob.setOutputValueClass(IntWritable.class);    //指定要处理的数据所在的位置  FileInputFormat.setInputPaths(wcjob, "hdfs://hdp-server01:9000/wordcount/data/big.txt");  //指定处理完成之后的结果所保存的位置  FileOutputFormat.setOutputPath(wcjob, new Path("hdfs://hdp-server01:9000/wordcount/output/"));    //向yarn集群提交这个job  boolean res = wcjob.waitForCompletion(true);  System.exit(res?0:1);  } |

### 5.2.2 程序打包运行

1. 将程序打包
2. 准备输入数据

vi /home/hadoop/test.txt

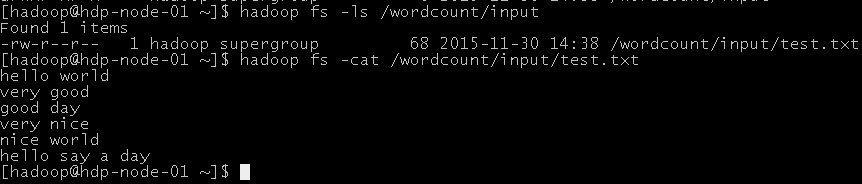
|  |
| --- |
| Hello tom  Hello jim  Hello ketty  Hello world  Ketty tom |

在hdfs上创建输入数据文件夹：

hadoop fs mkdir -p /wordcount/input

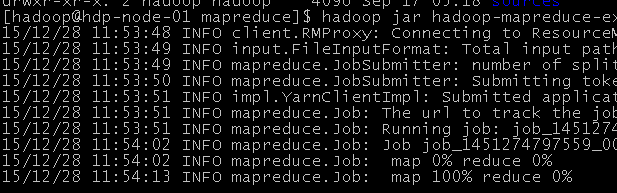
将words.txt上传到hdfs上

hadoop fs –put /home/hadoop/words.txt /wordcount/input



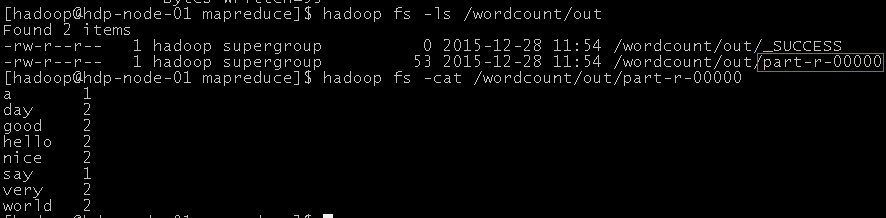
1. 将程序jar包上传到集群的任意一台服务器上
2. 使用命令启动执行wordcount程序jar包

$ hadoop jar wordcount.jar cn.itcast.bigdata.mrsimple.WordCountDriver /wordcount/input /wordcount/out



1. 查看执行结果

$ hadoop fs –cat /wordcount/out/part-r-00000



问题总结：

1/运行mr程序出错

connecting to resoucemanager

retrying ....

retrying .....

原因是没有启动yarn或者启动失败

2/初始化工作目录结构

hdfs namenode -format 只是初始化了namenode的工作目录

而datanode的工作目录是在datanode启动后自己初始化的

**3/datanode不被namenode识别的问题**

namenode在format初始化的时候会形成两个标识：

blockPoolId：

clusterId：

新的datanode加入时，会获取这两个标识作为自己工作目录中的标识

一旦namenode重新format后，namenode的身份标识已变，而datanode如果依然

持有原来的id，就不会被namenode识别

**4/datanode下线后多久看到效果**

datanode不是一下线就会被namenode认定为下线的，有一个超时时间

5/关于副本数量的问题

副本数由客户端的参数dfs.replication决定（优先级： conf.set > 自定义配置文件 > jar包中的hdfs-default.xml）